International Journal of Hybrid Information Technology
Vol.8, No.10 (2015), pp.299-310
http://dx.doi.org/10.14257/ijhit.2015.8.10.27

Discrete Particle Swarm Optimization Algorithm in Flexible Hybrid Flow Shop Scheduling

Liu Dongdong, Liu Kai, Zhao Zhengping Han Bo and Zhang Yan

Fuyang Normal College School of Computer and Information Engineering, Fuyang, Anhui Province, 236041
E-mail:ourliudd@163.com

Abstract

Traditional flexible flow shop scheduling cannot adapt to the work processes with existence of parallel machines, and blocks or limits the processes with no-wait constraints. Firstly, according to the problem in NWBFFSSP, which minimizes the maximum time used in the flow shop, an optimal solving model has been designed to realize the flexible flow shop scheduling with multi constraints; besides, for the distribution of machinery is improved, Finally, in the solving process, a real-time release priority strategy has been proposed to determine processing machine for each work piece. Furthermore, a methodology to detect work piece conflicts has been introduced while the conflicts are then eliminated by a kind of right moving strategy based on the maximum difference. The experimental results verify the effectiveness and feasibility of the proposed algorithm.

Keywords: *Flexible Flow Shop Scheduling, Blocking, No-wait, Discrete particle swarm optimization*

1. Introduction

Flexible Flow Shop Scheduling Problem (FFSSP) is extended from the traditional Flow Shop Scheduling Problem (FSSP). It features parallel running machines in several processes. Thus it is more suitable for actual manufacturing, and widely applied in chemistry, steelmaking, pharmacy, *etc*. Traditional FSSP assumes infinite large damping area between the adjacent machines of two processes. In actual scenarios, however, damping areas are always finite in size due to process requirement and product characters. Sometimes there will be no damping area at all. The scheduling problems are divided into two categories. One is the Blocking Flexible Flow Shop Scheduling Problem (BFFSSP). Due to the limit of damping areas, a work piece will be blocked on the machine even then process is completed, but the next machine is busy and the damping area is full. It will not be released until vacant machine is found in the following process, or the damping area gets freed. The other one is the No-Wait Flexible Flow Shop Scheduling Problem (NWFFSSP). In such situation, no pause is allowed in the entire process of one work piece. It will go through all processes without any stop once starts.

Article 1-6 proposed solutions to the BFFSSP. Article 7-12 proposed solutions to the NWFFSSP. They achieved satisfying results on those two standalone problems. However very few study cases were found aimed to the FFSSPs with both Blocking and No-wait factors. In article 13, the passenger trains were regarded as the work pieces with no-waiting restriction. The cargo trains were regarded as the ones with blocking restrictions. The NWBPMJSS_Liu_Kozan Algorithm was adopted to establish the Single train dispatching schedule when both passenger trains and cargo trains were present. No other research paper on the NWBFFSSP was found at this time. Business the batch issue of work piece is close to the actual production of the enterprise, it has been attracted the researches of domestic and foreign scholars in recent years. Juquan Yong, *et al.*, [4]

ISSN: 1738-9968 IJHIT

proposed a flexible batch algorithm. This algorithm take compression technology to combine the adjacent and same work pieces into the same sub-batch; Sun Zhijun, *et al.*, [5] proposes a novel genetic equivalent batch scheduling algorithm, which can determine the number of sub-grant. At the same time the arrangement of the order of the sub-grant processing is optimized; Bai Junjie, *et al.*, [6] proposes a flexible batch splitting method based on the "Cursor", which makes the making batch size adjusted according to the machine load; Zeng Qiang, *et al.*, [7] propose an wintness-based equivalent batch optimization method to optimize batch of processing products for scheduling equal tranches multi-objective optimization problem in parallel machine job shop; Martin, *et al.*, [8] introduce the heuristic rules to generate batch, and using the genetic algorithm make global search to the basic batch, thus batches results are optimizated.

This paper conducted a different study on the No-Wait Blocking Flexible Flow Shop Scheduling Problem (NWBFFSSP). A solution model was built for minimizing the max work time. The method used encoding based on permutation replacement to design the forward iterative algorithm for the target value. The discrete particle swarm optimization was adopted for overall optimization. The Iterated Greedy (IG) was adopted for enhancing the local searching capability of each individual. Both the First Release First and First Complete First strategies were used in deciding the best following machines. The detection of collision on work pieces was proposed, as well as the Right-movement relieving strategy based on the maximum difference. The effectiveness of the algorithm was validated through numerical simulation.

The innovation points of this paper:

(a)Traditional flexible flow shop scheduling is unable to realize the no-wait machining process of parallel machines, especially in the multi-constraint conditions. This paper aims to solve the flexible flow shop scheduling problem with the presence of congestion and no-wait constraints, and proposes a discrete particle swarm optimization method for the multi constrained flexible flow shop scheduling problem.

(b)In the realization process of discrete particle swarm optimization method for the problem, an optimal solving model is firstly designed according to the problem in NWBFFSSP, which minimizes the maximum time used in the flow shop. Besides, a forward iterative algorithm is proposed to calculate the target value while the permutation encoding is also applied. Finally, the discrete particle swarm optimization algorithm is utilized for global optimization. In the discrete particle swarm optimization algorithm, to avoid premature, an iterative greedy algorithm has been put forward to improve the local individual searching ability, and to realize the flexible flow shop scheduling in multi constraint conditions.

(c)Two machine distribution strategies: First Release First (FRF) and First Complete First (FCF) are proposed in the problem on distribution of machinery. A real-time release priority strategy is introduced to determine processing machine for each work piece. Furthermore, a methodology to detect work piece conflicts has been introduced while the conflicts are then eliminated by a kind of right moving strategy based on the maximum difference.

2. Problem Model

Let J_i be the i th work piece, $i=1,2,...,n$, n is the total amount of the pieces. π_j is the total number of parallel machines in process j, and $j=1,2,...,\lambda$; λ is the overall number of processes. m_k stands for the k th working machines and $k=1,2,...,m$. m is the total amount of standing machines. $S_{i,j}$ is noted as the start time of work piece J_i in process j. $t_{i,k}$ is the process time span of work piece J_i on machine m_k. R_k is the releasing time point of machine m_k. The time is changing dynamically. $C_{i,j}$ stands for

the closing time of work piece J_i on process j. The longest time of them is marked as $C_{\max}$, known as $C_{\max} = \max(C_{1,\lambda}, C_{2,\lambda}, ..., C_{n,\lambda})$. The general description of an NWBFFSSP is presented as following: A number of n work pieces is going through λ processes, their process restrictions are

There're at least one process contains parallel machines

Despite which machine one work piece is staying on, it always takes same time span to complete on specific process. All work pieces go through the same sequence of processes.

One process must be completed on single one machine

The max number of work pieces being processed on a machine is ONE

There's at least one process with the No-wait restriction, and one with the Blocking restriction

The final solution comes as a process sequence $\{J_1, J_2, ..., J_n\}$. The sequence minimizes the max working time, when the work pieces are processed in its order. The math model of NWBFFSSP in the paper is shown as following.

$$\min C_{\max} \tag{1}$$

$$\sum_{k=1}^{\pi_j} x_{i,j,k} = 1, i = 1,2,...,n;\ j = 1,2,...,\lambda \tag{2}$$

$$x_{i,j,k} = \begin{cases} 1 & work piece J_i \text{ use maching k in process j} \\ 0 & otherwise \end{cases} \qquad 0 \le \sum_{i=1}^{n} y_{i,k,t} \le 1, k = 1,2,...,m \tag{3}$$

$$y_{i,k,t} = \begin{cases} 1 & \text{at time t the machine } m_k \text{ is workin on piece } J_i \\ 0 & otherwise \end{cases}$$

$$C_{i,j} = S_{i,j+1}, i = 1,2,...,n;\ j = 1,2,...,\lambda - 1; u_i = 1 \tag{4}$$

$$S_{i,j} = \max(C_{i,j-1}, R_k), i = 1,2,...,n;\ j = 2,3,...,\lambda;$$

$$x_{i,j,k} = 1; u_i = 0 \tag{5}$$

$$u_i = \begin{cases} 1 & \text{work piece } J_i \text{ has the feature of No - wait restriction during process} \\ 0 & otherwise \end{cases}$$

Among them equation (1) is the solution/goal of the entire problem. Equation (2) states the restriction that one process must be completed on single one machine. Equation (3) gives that the max number of work pieces being processed on a machine is one. Equation (4) ensures there's no pause during the process of a work piece. Equation (5) puts on the Blocking restriction to certain pieces

3. Proposed Algorithm

A Initial conditions

The Advance - iterative algorithm was employed to arrange each work piece into the process sequences. Its initial condition was a given process sequence L of exchange codes. The sequence presented the order of pieces in first process. Assume all machines were available, which meant $R_k = 0, k = 1,2,...,m$, $C_{i,j} = 0, i = 1,2,...,n;\ j = 1,2,...,\lambda$, $Complete_j = 0,\ j = 1,2,...,\lambda$. $Complete_j$ stood for the process conditions of every work piece in process j.

$$Complete_j = \begin{cases} 1 & \text{All pieces are done in process } j \\ 0 & \text{Otherwise} \end{cases}$$

$Available_j$ stood for the group of all available machines in process j. The machines are sorted by their serial number in ascending. Its value range is shown in equation (6) and (7)

$$Available_j \in [1, \pi_j], j = 1 \tag{6}$$

$$Available_j \in [\sum_{l=1}^{j-1} \pi_l + 1, \sum_{l=1}^{j-1} \pi_l + \pi_j], j = 2,3,...,\lambda \tag{7}$$

B FRF Strategy and FCF Strategy

Both FRF and FCF strategies were adopted by the paper for choosing machines. The FRF focused on releasing machine m_k with earliest time in the incoming process j, to meet the requirement of equation (8).

$$m_k \wedge (R_k = \min R_l), k、 l \in Available_j \tag{8}$$

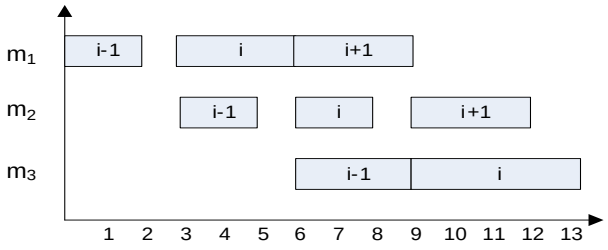
The FCF focused on minimizing the process of machine m_k, and meeting the requirements of equation (9).

$$m_k \wedge \min(C_{i,j} * x_{i,j,t}), k、 t \in Available_j, i = 1,2,...,n \tag{9}$$

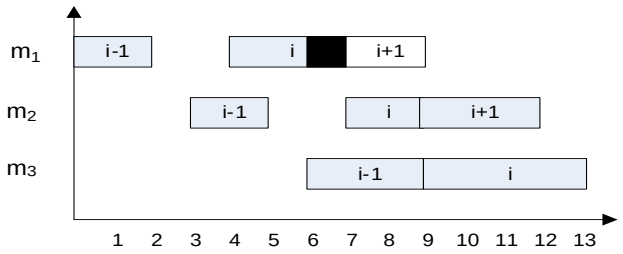
For the choice of work pieces, it is in accordance to the initial sequence given. In other following processes, the choices were made based on the minimum work time from last process. If the finish times of several pieces are same, one of them is randomly chosen.

C The Adjust Strategy of NWFFSSP

For the processes with No-wait restrictions, no pause is allowed. However the rule may be broken during the solution of this problem. Thus a pan-movement method based on differences was proposed in this paper. The method was used for the adjustment of both start and finish time of the NWFFSSP work pieces. The detailed procedures are : calculate the difference $\Delta = | C_{i,j} - S_{i,j+1} |$ of two adjacent processes j and $j+1$ of work piece J_i. If $\Delta \neq 0$, then pan the process time of J_i on process $1-j$ with a length of Δ units along the time axis. The adjustment on No-wait work pieces by differential pan-movements is shown in Figure 1. In fig 1 the work piece J_i is possessed with a No-wait restriction. In fig 1(a), the finish time of J_i in machine 2 is different from the start time of machine 3 downwards. The difference is Δ, which is against the restriction (4). The corresponding adjustment is shown in fig 1(b). By moving the process times of J_i in both machine 1 and 2 towards the right side of time axis, the J_i in Fig. 1(b) again meets the No-wait restriction.



(a)



(b)

Figure 1. The adjustment for NWFFSSP

D Conflict Detection and Removing

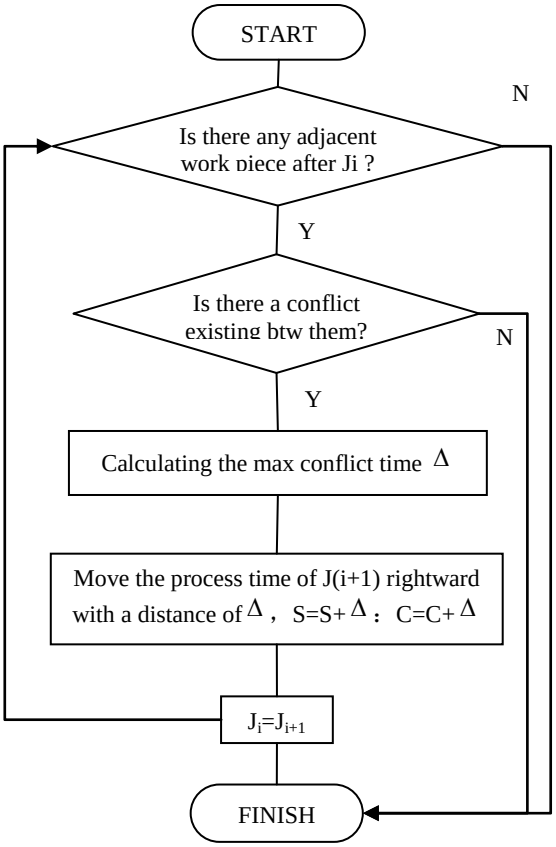


Figure 2. The Max Difference Strategy for Conflict Removing

E Advance – Iterative Algorithm

The advance – iterative algorithm was employed for choosing work pieces and assigning them to machines. The major idea is following. The work piece will be assigned to a specific process $j(j=1,2,...,\lambda)$, if there're available machines in the process. Otherwise move to next process until any machine is available in process $j'(j'>j)$. At the same time the work piece is released from the machine in process $j'-1$. The algorithm starts with iteration from process j, looking for available machines. The work piece will then be released from previous machine, and assigned to the newly found one. Work pieces are assigned to following processes with each advance, and assigned to previous processes with each iteration. By repeating the procedure, every work piece is processed with minimum time cost.

The procedure of running Advance – iterative algorithm is presented as following.

Step 1 Assign the work piece to first process. If $\pi_1 = 1$, assign piece L_1 to machine m_1. If $\pi_1 > 1$, choose a total number of π_1 work pieces in sequence from L, and assign them to machine $[m_1, m_{\pi_1}]$. The major rule is to minimize the process time of a piece on the specific machine.

Step 2 Alter the initial conditions. $L = L - L_i, i = 1,2,...,\pi_1$, $Available_1 = \phi$.

Step 3 Execute step 4 when $j = 1,2,...,\lambda$.

Step 4 If $Complete_j = 0$, go to step 5; otherwise $j = j + 1$. When $j > \lambda$, jump to step 11; otherwise repeat step 4.

Step 5 If $Available_j = \phi$, no available in current process. Go to step 6. Otherwise process is available in current stage.

Step 5.1 If $j = 1$, Choose work piece L_1 from sequence L with FCF or FRF strategy. Assign the piece to vacant machine m_k in $Available_j$. When $L = L - L_1$, $Available_j = Available_j - m_k$, $C_{L_1,j} = R_k + t_{L_1,j}$, $S_{L_1,j} = C_{L_1,j} - t_{L_1,j}$, go back to step 5; otherwise go to 5.2.

Step 5.2 When $j \neq 1$, choose all work pieces J_i meeting the condition $C_{i,j-1} = \{\min(C_{t,j-1}) \wedge C_{t,j-1} \neq 0\}, t = 1,2,...,n$. Assign them to the available machines m_k in $Available_j$ with strategy FCF or FRF. $Available_j = Available_j - m_k$, $Available_{j-1} = Available_{j-1} + m_{k'}$. $m_{k'}$ stands for the machine J_i was processed with in order. $C_{i,j} = \max(R_k, C_{i,j-1}) + t_{i,k}$, $S_{i,j} = C_{i,j} - t_{i,k}$, $R_{k'} = S_{i,j}$. $u_i = 0$ means a No-wait restriction occurred on piece J_i. Mark $\beta = j$ and jump to step 8, otherwise return back to step 5.

Step 6 Let δ be the following processes of j, execute step 7 for $\delta = j+1, j+2,...,\lambda$.

Step 7 When $Available_\delta = \phi$, let $\delta = \delta + 1$ and jump to step 6. Otherwise choose all work pieces J_i meeting the condition $C_{i,\delta-1} = \{\min(C_{t,\delta-1}) \wedge C_{t,\delta-1} \neq 0\}, t = 1,2,...,n$, Assign them to the available machines m_k in $Available_\delta$ with strategy FCF or FRF. The $Available_\delta = Available_\delta - m_k$, $Available_{\delta-1} = Available_{\delta-1} + m_{k'}$, $m_{k'}$ is the machine that J_i stayed on in process $\delta - 1$. $C_{i,\delta} = \max(R_k, C_{i,\delta-1}) + t_{i,k}$, $S_{i,\delta} = C_{i,\delta} - t_{i,k}$, $R_{k'} = S_{i,\delta}$. Let $\beta = \delta$. When $u_i = 0$, J_i has a No-wait restriction in its processes. Then jump to step 8, otherwise step 10.

Step 8 Adjust the start and finish times of each work piece with No-wait restriction, with the reference of equation (4). Let $\Delta = S_{i,\beta} - C_{i,\beta-1}$. When $\Delta > 0$, the regulation of equation (4) is violated, apply necessary adjustments from chapter 2.3.

Step 9 Detect any conflicts. Remove the conflicts with rightward pan-movement with based on max differences.

Step 10 When $\beta = \lambda$, let $R_k = C_{i,\lambda}$, let $Available_\lambda = Available_\lambda + m_k$. Then return back to step 4 and start looking for another work piece.

Step 11 The algorithm ends. The final result is exported.

F *Particle Code*

The permutation coding mechanism is utilized in this algorithm. Each individual in the population, which is a work piece in the work pieces sequence, represents a solution. For example, an individual L is encoded as $L=\{3,6,1,2,4,5\}$, which means there are six work pieces in total while the sequence of the work pieces in the first work process is 3, 6, 1, 2, 4 and 5. That means that the six work pieces are processed following the sequence, i.e. the 3rd work piece are processed first, and then followed by the 6th work piece, and so on.

In order to obtain larger search space, work piece sequences are randomly generated for each individual in the procedure of population generation. Forward iterative algorithm is used for decoding to calculate the fitness value of individual.

G Particle Updating Method

Assume $P_i(k)$ to be the optimal solution of the ith particle after k iterations, while the current optimal solution for all particles are noted as $P_g(k)$ .Because the decoding is discrete, the speed and position are hard to describe if following basic particle swarm optimization algorithm。Therefore, the modify speed, position updating formula are defined as (11) and (12)，where $\otimes$ represents crossover operation.

$$V_i(k+1)=V_i(k)\otimes P_i(k)\otimes P_g(k) \quad (11)$$

$$X_i(k+1)=X_i(k)\otimes V_i(k+1) \quad (12)$$

This paper uses the method of Partially Mapped Crossover as the crossover operation in the particle updating formula. For two individuals θ_1 and θ_2, the crossover part is randomly selected. The crossover part of θ_2 is moved to replace the corresponding part of θ_1. The former content in the part of θ_1 is deleted, while the corresponding mapping replacements are completed in this step.

In order to avoid falling into local convergence, the fitness values of the updated particles are compared with the former values. If the updated particles are with better fitness values, they will replace the individual maximum $P_i(k)$. Otherwise, $P_i(k)$ will keep unchanged for several generations, which will then cause new search on the neighborhood of the particles. Similarly, if $P_g(k)$ keeps unchanged for several generations, which indicates the particle swarm has stopped, neighborhood search will be conducted in the population 'global maximum' $P_g(k)$, which will force the particle swarm jump out of local convergence.

This paper applies iterated greedy algorithm on neighborhood search for particle individual maximum and population global maximum.

Step 1 Generation of initial solution S_0
While(End condition is not satisfied) Do
Step 2 Destruction operation $S_p = Destruction(S_0)$。
Step 3 Construction operation $S'□ Construction(S_p)$
Step 4 $S_0 receive(S_0, S')$
End

H Flow Chart of the Algorithm

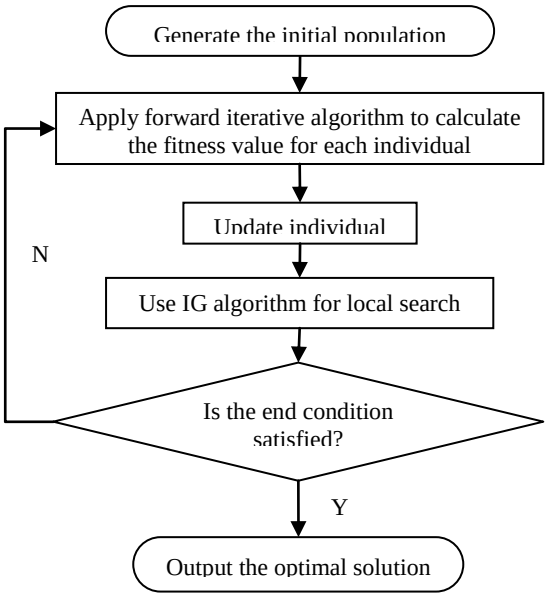


Figure 3. Flow Chart of the Algorithm to Solve and Optimize NWBFFSSP

4. Experimental Results

The algorithm is simulated by VC++6.0, and conducted in a PC with CPU of Pentium 2.10GHz, RAM of 2.0GB. The key parameters are $popsize = 50$, $gen = 10000$. To evaluate its performance, three different examples are used for the evaluation.

A Experiment on Feasibility Analysis

Data in example 1 comes from [15], which includes 12 work pieces, 9 machines, and 3 work processes. There are 3, 2, and 4 parallel machines in each work process.

The algorithm uses the FRF strategy and FCF strategy in the selection of processing machine. Each strategy runs for 10 times. Work pieces J_0、J_2、J_4、J_7、J_9 are randomly selected as the work pieces without constraints while other work pieces are with constraints. The result is shown in Table 1.

In Table 1, left numbers in the divide are the values of $C_{\max}$, while the right numbers presents time (Units). From this table, the optimal solutions by FRF and FCF are 24. However, the time consumption of FRF and FCF are 10.3s and 4.3s respectively. This shows that FCF performs better than FRF. The Gantt chart of this solution can be found in Figure 4.

Table 1. The Result of the Algorithm in Example 1

Time	1	2	3	4	5
FRF Strategy	24/6	24/5	24/1	24/15	24/25
FCF Strategy	24/1	24/9	24/12	24/10	24/1
Time	6	7	8	9	10
FRF Strategy	24/7	24/1	24/25	24/16	24/2
FCF Strategy	24/2	24/2	24/2	24/2	24/2

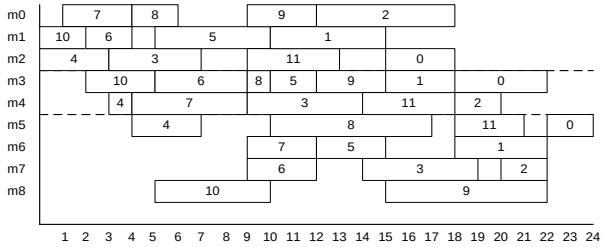


Figure 4. Gantt Chart of the Optimal Solution

To further verify the performance of the proposed algorithm, it's compared with EDA[15], SFLA[16] and GA[17] for solving flexible flow-shop scheduling problem. Although the problem in this section is more complex, and with stranger constraints, it has a certain reference function to be used for comparison with EDA, SFLA and GA. Comparison results are shown in Table 2.

From Table 2, solution by the proposed algorithm is similar with that of SFLA, better than GA, but a little worse than EDA. Therefore, it can be said that the proposed algorithm has better performance in solving NWBFFSSP.

Table 2. Comparison of the Results from Different Algorithms based on Example 1

Time	1	2	3	4	5
SFLA	24	24	24	24	24
EDA	23	24	23	23	23
GA	30	27	26	27	29
The proposed algorithm	24	24	24	24	24
Time	6	7	8	9	10
SFLA	24	24	24	24	24
EDA	23	24	24	23	24
GA	27	26	27	26	28
The proposed algorithm	24	24	24	24	24

B Experiment on Efficiency Analysis

Example 2 is from literature [15] with large scale data. In this example, there are 12 work pieces, 10 machines, and 4 work processes. The sequence of the machines is 3, 3, 2, 2.

The algorithm randomly selects J_0、J_2、J_4、J_7、J_9 to be without any constraints, while other work pieces are with delay constraints. FRF and FCF strategy are used in the selection of processing machine, and independently operated for 10 times. Figure 5 shows the Gantt chart of the optimal solution, which is 298.

Similar with example 1, the proposed algorithm is compared with EDA, SFLA and GA, which is shown in Table 3. Literature [17] also presents a simulation result, readers can find more information if they are interested.

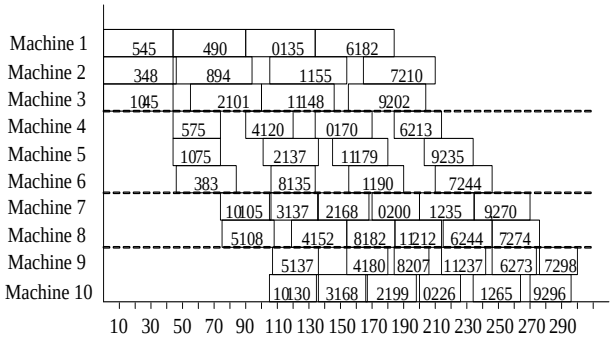


Figure 5. Gantt Chart of the Optimal Solution in Example 2

Table 3. Comparison of the Results by Different Algorithms in Example 2

Time	1	2	3	4	5	6	7	8	9	10
GA					347					
SFLA	297	313	297	297	313	310	313	311	316	306
EDA	297	297	297	297	298	297	297	298	298	298
The proposed algorithm	298	298	298	298	298	298	298	298	298	298

Time	1	2	3	4	5
GA			347		
SFLA	297	313	297	297	313
EDA	297	297	297	297	298
The proposed algorithm	298	298	298	298	298
Time	6	7	8	9	10
GA			347		
SFLA	310	313	311	316	306
EDA	297	297	298	298	298
The proposed algorithm	298	298	298	298	298

From Table 3, the optimal algorithm in this paper is better than GA, but a litter poorer than SFLA and EDA. The reason behind this result is that the structure of this algorithm is total different from that in algorithms in [15] and [16], which causes more free space in NWBFFSSP. On the other hand, the proposed algorithm obtains a solution of 298, which is much better than 307.3 of SFLA. In general, the performance of this algorithm is better than GA algorithm and SFLA algorithm, but a little worse than EDA algorithm.

From the experiments above, it's declared that the proposed algorithm is feasible to solve NWBFFSSP, and achieves high quality solution through the comparison with other.

5. Conclusion

The method of solving NWBFFSSP has been firstly proposed in this paper. The initial work piece processing sequence is generated by permutation coding algorithm of particle swarm optimization. Then the forward iterative algorithm is applied to decode and calculate the target value. In the calculation, the machines selected by the FCF strategy and FRF strategy are used, and a right moving strategy based on the maximum difference is introduced to eliminate the conflicts. Local searching capability of particles has been enhanced by IG algorithm. Numerical simulations based on examples in solving NWBFFSSP have verified the feasibility and the effectiveness of the proposed method.

ACKNOWLEDGEMENTS

The work is supported by The National Natural Science Fund(No.61401101) and The natural science foundation of Anhui Province (No.1408085QF122) and The National College Students' innovative projects (No.201410371030, No.AH201410371037) and The national statistical science project plan research (No.2014LZ32) and Fuyang Normal College Foundation (No.2013KJFH05, 2013FSKJ14) and Anhui Province Quality Project (No.2014zy048)

And The Education Department of Anhui Province Natural Science Fund (No.KJ2013B148)

References

[1] M. S. Salvador, "A solution to a special case of flow shop scheduling problems", Symposium of the Theory of Scheduling and Applications. New Work: Springer, **(1973)**, pp. 83-91.
[2] C. Lu, Y. Lifeng and C. Jianguo, "A solution with a blocking restrictions Hybrid Flow Shop search algorithm", Shanghai Jiaotong University, vol. 40, no. 5, **(2006)**, pp. 856-859.
[3] Z. Lv and T. Su, "3D seabed modeling and visualization on ubiquitous context", In SIGGRAPH Asia Posters, ACM, **(2014)**, p. 33.
[4] Z. Lv, L. Feng, S. Feng and H. Li, "Extending Touch-less Interaction on Vision Based Wearable Device", Virtual Reality (VR), IEEE, **(2015)**.
[5] T. M. Reza, S. Nima and S. Farrokh, "A memetic algorithm for the flexible flow line scheduling problem with processor blocking", Computers & Operations Research, vol. 36, no. 2, **(2009)**, pp. 402-414.
[6] Y. Kun, S. Nathalie and S. Christophe, "Application of EM algorithm to hybrid flow shop scheduling problems with a special blocking", Proceedings of the 14th IEEE International Conference on Emerging Technologies and Factory Automation, Mallorca, USA, IEEE, **(2009)**, pp. 1-7.
[7] X. Hua and T. Lixin, "HFS scheduling real-time without waiting for a Lagrangian relaxation algorithm", Control and Decision, vol. 21, no. 4, **(2006)**, pp. 376-380.
[8] L. Jianxiang, T. Lixin and W. Huijiang, "Transport and setup time with no-wait flow shop scheduling problem in parallel", Systems Engineering Theory and Practice, vol. 26, no. 1, **(2006)**, pp. 18-25.
[9] L. Yan and L. Tieke, "Based on Constraint Programming without waiting Hybrid Flow Shop Scheduling Problem", Process Automation Instrumentation, vol. 34, no. 3, **(2007)**, pp. 26-29.
[10] J. Fariborz, S. Shaya and R. Massoud, "A genetic algorithm for solving no-wait flexible flow lines with due window and job rejection", Int J Adv Manuf Technol, vol. 42, no. 1-2, **(2009)**, pp. 523-532.
[11] J. M. Garcia and S. Lozano, "Production and delivery scheduling problem with time windows", Computers & Industrial Engineering, vol. 48, no. 5, **(2005)**, pp. 733–742.
[12] S. Jiwei and T. Jiafu, "DPSO-based no-wait Hybrid Flow Shop Scheduling", Journal of System Simulation, vol. 22, no. 10, **(2010)**, pp. 2257-2261.
[13] Y. Geng, Y. Wan, J. He and K. Pahlavan, "An Empirical Channel Model for the Effect of Human Body on Ray Tracing", 2013 IEEE 24nd International Symposium on Personal Indoor and Mobile Radio Communications (PIMRC), London, Britain, **(2012)** September, pp. 47-52.
[14] Y. Geng, J. Chen and K. Pahlavan, "Motion detection using RF signals for the first responder in emergency operations: A PHASER project", IEEE 24nd International Symposium on Personal Indoor and Mobile Radio Communications (PIMRC), London, Britain, **(2013)** September.
[15] S. Q. Liu and K. Erhan, "Scheduling Trains with Priorities:A No-Wait Blocking Parallel-Machine Job-Shop Scheduling Model", Transportation Science, vol. 45, no. 2, **(2011)**, pp. 175-198.
[16] R. Rubén and S. Thomas, "A simple and effective iterated greedy algorithm for the permutation flow shop scheduling problem", European Journal of Operational Research, vol. 177, no. 3, **(2007)**, pp. 2033-2049.
[17] W. Shengrao, W. Ling and X. Ye, "Hybrid Flow Shop Scheduling Problem Solving Estimation of Distribution Algorithms", AAS, vol. 38, no. 3, **(2012)**, pp. 437-443.
[18] Z. Huiren, T. wansheng and W. Yinghui, "Flexible Flow-Shop Scheduling genetic algorithm", Computer Engineering and Applications, vol. 45, no. 30, **(2009)**, pp. 224-226.

Authors

LIU Dongdong, received his M.S. degree in Computer application from Dalian Maritime University in Dalian, China. He is currently a lecturer in the Fuyang Normal College. His research interest is mainly in the area of Computer Software, Network, Algorithm optimization.