Fireworks Explosion Algorithm for Hybrid Flow Shop Scheduling and Optimization Problem

Dong Xiao Ting\textsuperscript{1,a}, Liu Chang Ming\textsuperscript{1}, Han Zhong Hua\textsuperscript{2}

\textsuperscript{1}Department of Electrical Engineering, Sichuan College of Architectural Technology, De'yang, Sichuan 618000, China
\textsuperscript{2}Department of Digital Factory, Shenyang Institute of Automation, Chinese Academy of Sciences, Shenyang, Liaoning110016, China
\textsuperscript{a}Email: dxt199211@163.com

Abstract: fireworks explosion algorithm is a relatively new population intelligent evolution algorithm, having a good performance in local mining and global search for optimal solution while little weak in searching speed. Therefore, this study focus on researching how to use the fireworks explosion algorithm (FEA) to deal with hybrid flow shop scheduling and optimization problem effectively both in accuracy and speed. The main contribution of this study is to develop a new firework individual encoding and decoding methods which could help FEA to solve HFS problem easily and effectively. Several practical instances testing proved that the FEA algorithm has better performance in dealing with HFS problem than GA and CS algorithm.

1. Introduction

The fireworks explosion algorithm (FEA) [1] is a recently developed swarm intelligent evolution algorithm, inspired by the real explosion process of fireworks. The main idea of the FEA is to use the explosion of the fireworks to search the feasible space of the optimization function, which is a brand new search manner. At present, the fireworks algorithm has been applied to many practical optimization problems [2], the application areas include the factorization of a non-negative matrix [3], multi-objective optimization of a multimodal transportation network problem [4], travelling salesman problem [5], the controllers structure and parameters selection problem [6], warehouse-scheduling problem [7], the parameter estimation of chaotic systems [8], the scheduling of multi-satellite control resources [9], etc.

Hybrid flow shop scheduling and optimization problem (HFS) is a typical NP-hard problem, has a big industrial background than traditional flow shops, like glass, steel, car and textile industries. HFS scheduling problem are more complicated for it not only includes how to make sequences to all jobs, but also contains how to assign jobs in parallel machines at each stage. So, it is of strong academic significance and engineering application value to study the HFS problems. From the current literature, one can easily find that FEA algorithm has been widely used to solve optimization problems, however, the application of FEA algorithm to solve scheduling problem is limited [10-11]. In this paper, firework explosion (FEA) algorithm is presented to minimize makespan for the HFS scheduling problems.
2. Problem definition
The HFS scheduling problem can be described as follows: (1) there are \( n \) jobs which will be processed; (2) each job must experience \( m \) stages with the same direction; (3) there has at least one machine at each stage; (4) there exists at least one stage which has more than one machine; (5) machines in each same stage are unrelated; (6) each job can be assigned to any of the \( M_j \) \( (M_j \geq 1) \) machines at stage \( j \) \( (j = 1, 2, ..., m) \); when the former stage is completed, the job will join in the waiting queue of buffer area which is infinite and waiting entering the next stage. The objective is to minimize the makespan \( (C_{\text{max}}) \). The makespan for a scheduling problem is defined as the completion time of the last job to leave the production system.

2.1. Notion
\( n \), represent the number of jobs waiting to be processed.
\( m \), represent the number of stages which each job needs to be processed.
\( k \), represent the machine on which job \( i \) being operating.
\( M_j \), represent the maximum machine number of stage \( j \).
\( j-k \), represent the \( k^{\text{th}} \) machine at stage \( j \), \( j \in \{1, 2, ..., m\} \), \( k \in \{1, 2, ..., M_j\} \).
\( S_{ij} \), represent the starting time of job \( i \) at stage \( j \).
\( C_{ij} \), represent the end time of job \( i \) at stage \( j \).
\( P_{ij} \), represent the processing time of job \( i \) at stage \( j \).

2.2. Variables and Constraint
\( Y_{ik} \), a binary variable which is equal to 1 if job \( i \) is assigned to machine \( k \) at stage \( j \), otherwise equal to 0.
\( Z_{ip} \), a binary variable which is equal to 1 if job \( i \) is processed on preferential position, otherwise equal to 0.

The HFSP can be formulated as follows:
Minimize: \( C_{\text{max}} \) \hspace{1cm} (1)
\[ \sum_{i=1}^{n} Z_{ip} = 1, i \in \{1, 2, ..., n\} \tag{2} \]
\[ \sum_{p=1}^{n} Z_{ip} = 1, p \in \{1, 2, ..., n\} \tag{3} \]
\[ \sum_{k=1}^{M_j} Y_{ik} = 1, i \in \{1, 2, ..., n\}, j \in \{1, 2, ..., m\} \tag{4} \]
\[ C_{ij} = S_{ij} + P_{ij} \tag{5} \]
\[ S_{ij} > C_{i,j-1}, j = 2, 3, ..., m \tag{6} \]

In above constraints, formulas (2)-(6) describe the normal HFSP model inherent constraint. Formula (2) and (3) ensure that each preferential position can be assigned to one job and each job can only has one preferential position; constraint (4) shows that each job can only be processed in one of the parallel machine in each stage; constraint (5) expressed the relationship between the starting time and the completion time of job \( i \), no matter at which stage it is; constraint (6) ensures that a job can’t be processed at next stage until it has finished being processed in the current one.

3. Fireworks Explosion Algorithm Description
Firework explosion algorithm is a new swarm intelligent evolution algorithm which possess special
optimal solution searching mechanism. In firework explosion algorithm, each spark can be regarded as an effective solution. Neighborhood search process is the process of generating a certain amount of sparks. The explosion Radius and numbers of each firework is different. Fireworks which possess superior fitness have smaller explosion radius, which could contribute to excavate the local searching ability. Instead, it will possess a bigger explosion radius if the firework has a poorer fitness, so that to expand global searching ability. Fireworks can exchange information and source with each other according to their fitness in order to make a balancing between the global searching and local searching. Furthermore, Gauss mutation sparks are brought in to strengthen the diversity of the solution population and enhance the ability to jump out of local extreme value. The process of firework explosion algorithm is as follows:

Step1: randomly generating an initial population which size is set as 30, \( N_i (i = 1, 2, ..., 30) \) represent the \( i^{th} \) individual which is a \( m \times n \) dimensional integer array in which the elements \( N_{i_c} \) means job \( c \) will be processed on parallel machine \( N_{i_c} \) in stage \( i \).

Step2: for \( i = 1 \) to 30, calculating fitness \( f(N_i) \) for each firework.

Step3: calculating the explosion radius \( R_i \) and sparks \( S_i \) for firework \( N_i \) according to formula (7) and (8).

\[
S_i = SN \bullet \frac{y_{\text{max}} - f_i + \delta}{\sum_{i=1}^{N} (y_{\text{max}} - f_i) + \delta} 
\]

(7)

\[
R_i = A \bullet \frac{f_i - y_{\text{min}} + \delta}{\sum_{i=1}^{N} (y_{\text{max}} - f_i) + \delta} 
\]

(8)

\[
S_i = \begin{cases} 
\text{round}(a \cdot SN), & S_i < a \cdot SN; \\
\text{round}(b \cdot SN), & S_i < b \cdot SN; \\
\text{round}(S_i), & \text{others};
\end{cases} 
\]

(9)

In the formula, \( SN \) is the amounts of basic explosion sparks between 5 to 30, \( A \) is the basic explosion radius which always between 10 to 30, \( y_{\text{min}} \) is the minimum fitness of the current population while \( y_{\text{max}} \) is the maximum, \( \delta \) is the machine minimum using to avoid division by zero. To ensure \( S_i \) to be an integer, a rounding rule is used. Formula (9) is to make the amount of explosion sparks vary in a relatively reasonable range.

Step4: generating \( S_i \) sparks based on the calculated value \( R_i \), the specific procedure is following: step1: for \( k = 1 \) to \( S_i \), using formula (10) to randomly generating a sequences \( SP \) whose size is equal to current individual \( N_i \); step2: making a position offset according formula (11) to produce a new individual \( N_{i-k} \); step3: if \( N_{i-k} \) is out of the boundary, using formula (12) to make it mapping to a new location. Step 4, calculating the fitness of \( N_{i-k} \), if \( f(N_{i-k}) \neq f(N_i) \), \( N_{i} = N_{i-k} \), \( f(N_{i}) = f(N_{i-k}) \); Step5, if \( k = S_i \), go to step6; otherwise, \( k = k + 1 \) and go to step1.

\[
SP = R_i \cdot \text{unifrnd}(-1, 1, m, n) 
\]

(10)

\[
N_{i-k} = N_i + SP 
\]

(11)

\[
N_{i-k} = LB + N_i (UB - LB) 
\]

(12)

In formula (11), (12) and (7): \( SP, N_i, N_{i-k}, LB \) and \( UB \) are arrays whose size is \( m \times n \); \( \text{unifrnd}(-1, 1, m, n) \) is a function that can generate an \( m \times n \) array whose elements value belong to \((-1, 1)\).
Step 5: generating a number \( r \) by \( \text{unifrnd}(-1,1) \), if \( r > R_a \), make a Gauss mutation by formula (13); if \( N'_i \) is out of the boundary, using formula (12) to deal with.

\[
N'_i = N_i + \text{normrnd}(-1,1,m,n) \tag{13}
\]

Step 6: Calculating the fitness \( f\left(N'_i\right) \), if \( f\left(N'_i\right) < f(N_i) \), then \( N_i = N'_i\). \( f(N_i) = f(N'_i) \).

Step 7: if \( i = 30 \), exit and output the minimize fitness value \( f_{\text{min}} \) using formula (14); otherwise, \( i = i + 1 \) and go to step 3.

\[
f_{\text{min}} = \min(f_{N_1}, f_{N_2}, \ldots, f_{N_{30}}) \tag{14}
\]

4. Computational Result

In this section, a comparison between the FEA, GA and CS was implemented based on two different practical instances. All algorithm is carried on MATLAB2012b, running on the PC with windows 7 system, corei7 CPU2.4HZ and 4G memory. There are 4 vital parameters in FEA: population size \( N_P \), the amount of basic explosion sparks \( SN \), basic explosion radius \( A \) and Gauss mutation sparks’ amount \( GN \). Since the parameters value has great impact on the preference of FEA, a reasonable parameters’ configuration is made as follows: \( N_P = 5 \), \( SN = 25 \), \( A = 30 \), \( GN = 50 \).

| Test no | 1  | 2  | 3  | 4  | 5  | 6  | 7  | 8  | 9  | 10 | Running time(s) |
|---------|----|----|----|----|----|----|----|----|----|----|-----------------|
| GA      | 29 | 29 | 28 | 27 | 28 | 28 | 28 | 28 | 30 | 27 | 3.329           |
| CS      | 28 | 30 | 30 | 26 | 29 | 27 | 27 | 28 | 26 | 26 | 3.096           |
| FEA     | 26 | 26 | 24 | 26 | 24 | 24 | 24 | 24 | 25 | 26 | 4.436           |

Instance 1 comes from literature [12], has 12 jobs and 3 stages, and the numbers of machines in each stage are 3, 2 and 4. The results are listed in Tables 1. Fig.2 illustrates the evolving process of the FEA based algorithm for solving instance 1.

It can be seen from table I that the FEA algorithm achieved the best solution 24, improved 11.1% and 7.69% compared to GA and CS algorithm respectively. Observing the evolve trend of the three algorithm, it’s evident that the FEA had a better evolve energy and can jump out of local minima in time.
Problem 2 comes from literature [13], has 12 jobs and 4 stages, and the numbers of machines in each stage are 3, 3, 2 and 2. The results are listed in Table 2. Fig.2 illustrates the evolving process of the SACS based algorithm for solving Problem 3.

| Test no | 1   | 2   | 3   | 4   | 5   | 6   | 7   | 8   | 9   | 10  | Running time(s) |
|---------|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|----------------|
| GA      | 325 | 327 | 325 | 327 | 327 | 328 | 327 | 327 | 327 | 327 | 5.410          |
| CS      | 313 | 314 | 313 | 313 | 317 | 314 | 313 | 313 | 313 | 313 | 5.309          |
| FEA     | 298 | 298 | 299 | 298 | 298 | 300 | 298 | 299 | 299 | 299 | 5.781          |

From Table II, it’s obvious that the FEA algorithm obtained the best solution 298. Analyzing the evolution trends of the three algorithms, it can be concluded that the FEA could keep evolution energy all around the evolution process, but little slowly to search the optimal solution.

5. Conclusion
In this paper, the main work studied is how to apply FEA to hybrid flow shop scheduling and optimization problems and testing the optimization efficiency. The computational results show that the FEA have a better global search ability and can jump out of local minima easier than GA and CS algorithm, and can solve the HFS problem effectively. There’s may two main reasons: (1) unique evolutionary mechanism in which the new sparks numbers and explosion radius depend on the fireworks fitness, which can avoid redundant and poor sparks and improve the efficiency of searching the optimal solution. (2) The superiority of Gauss mutation strategy which could help to keep evolution energy all the time and strengthen the ability to escape the local extreme value. The defect of the FEA is the slowly speed when evolving. So the next study field is how to accelerate the evolution speed of FEA.

Acknowledgment
Natural Science Project of the Education Department of Sichuan Province.

Reference
[1] Tan Y, Zhu Y. Fireworks algorithm for optimization [J]. Advances in swarm intelligence, 2010:
355-364.

[2] Tan Ying, Zheng Shaoqiu. Recent advances in fireworks algorithm [J]. CAAI Transactions on Intelligent Systems, 2014, 9(05):515-528.

[3] Janecek A, Tan Y. Using population based algorithms for initializing nonnegative matrix factorization [J]. Advances in swarm intelligence, 2011: 307-316.

[4] Mnif M, Bouamama S. Firework Algorithm For Multi-Objective Optimization Of A Multimodal Transportation Network Problem [J]. Procedia Computer Science, 2017, 112(C): 1670-1682.

[5] Taidi Z, Bennameur L, Chentoufi J A. A fireworks algorithm for solving travelling salesman problem [J]. International Journal of Computational Systems Engineering, 2017, 3(3): 157-162.

[6] Lapa K, Cpałka K. On the application of a hybrid genetic-firework algorithm for controllers structure and parameters selection[C]/Information Systems Architecture and Technology: Proceedings of 36th International Conference on Information Systems Architecture and Technology–ISAT 2015–Part I. Springer International Publishing, 2016: 111-123.

[7] Ye S, Ma H, Xu S, et al. An effective fireworks algorithm for warehouse-scheduling problem [J]. Transactions of the Institute of Measurement and Control, 2017, 39(1): 75-85.

[8] Li, H.; Bai, P.; Xue, J.; Zhu, J. Zhang, H. Parameter estimation of chaotic systems using fireworks algorithm. In Advances in Swarm Intelligence; Springer: Berlin, Germany, 2015: 457–467.

[9] Liu Z.B, Feng Z.R., Ke, L.J. Fireworks algorithm for the multi-satellite control. In Proceedings of the 2015 IEEE Congress on Evolutionary Computation, Sendai, Japan, 25–28 May 2015:1280–1286.

[10] Bao Xiaoxiao, YE Chunming, Huang Xia. Solving job-shop scheduling problem by firework algorithm. Computer Engineering and Applications [J], 2017, 53(3): 247-252.

[11] Lei C, Chunming Y, Xia H. Applying chaotic fireworks algorithm in solving permutation flow shop scheduling problem [J]. Computer Applications and Software, 2016, 33(11): 288-292.

[12] Wang J P, Zhu H J, Zhou J. Optimal sub-population genetic algorithm for flexible flow shop scheduling problem [J]. Application Research of Computers, 2012, 29(2): 442-444.

[13] Cui J S, Li T K, Zhang W X. Hybrid flow shop scheduling model and its genetic algorithm[J].Journal of University of Science and Technology Beijing, 2005, 27(5): 623-626.