An effective DE algorithm for hybrid flow shop load balancing scheduling problem

Han Zhonghua
Faculty of Information and Control Engineering, Shenyang Jianzhu University, Shenyang, China
Department of Digital Factory, Shenyang Institute of Automation, CAS, Shenyang, China

Dong Xiaoting, Lin Shuo
Faculty of Information and Control Engineering, Shenyang Jianzhu University, Shenyang, China
dxt199211@163.com

Abstract: In this paper, an improved DE algorithm called double population self-adaptive differential evolution (DPSADE) algorithm is used to solve the hybrid flow shop load balancing scheduling problem (HFS-LBSP) which combines a novel double population cooperative evolution mechanism with a special self-adaptive parameter adjusting strategy. A mathematical model in which the weighted summation of the processing time load balancing cost and the total parallel machine waiting time is set as load balancing comprehensive evaluation index is formulated for this problem. In the last, The comparison between DPSADE and DE, SADE demonstrate the effectiveness of DPSADE in solving HFS-LBSP.

Keywords: HFS scheduling problem; load balancing; selection probability; DE algorithm; DPSADE algorithm

I INTRODUCTION

The workshop load balancing scheduling problem was first conducted by Thomopoulos N T[1], much progress has been made ever since then. Researchers mainly focus on permutation flow shop, flexible job shop and parallel machine shop[2-5] while little literature concentrate on HFS-LBSP, literature[6] pointed out that keep load balancing in parallel machine is the key to improve the level of manufacturing performance, and a neighborhood search-based method is used together with genetic algorithm to solve this problem. Literature[7] made the load balancing as a constraint, and designed a artificial immune algorithm to balance the processing time on parallel machine in different stages, mainly to optimize the makespan problem. Hence, in this paper, a mathematical model is established in which a comprehensive evaluation index which takes processing time load balancing cost and the total parallel machine waiting time into consideration at the same time was designed, together with DPSADE algorithm to conduct a deeply research on HFS-LBSP.

II FORMULAITON OF HFS

The HFS scheduling problem can be described as follows: (1) there are n jobs which will be processed; (2) each job will experience m stages in the same direction; (3) there has at least one machine at each stage; (4) there are at least one stage which include more than one machines; (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 the buffer area and wait to enter the next stage. The objective of the scheduling is to balance the load on every parallel machine in each stage.

A Model Parameter

\(n\) is the number of jobs waiting to be processed.

\(J_i\) is the \(i^{th}\) job, \(i \in \{1, ..., n\}\);

\(m\) is the number of stages which each job must be processed.
$M_j$ is the maximum machine number of stage $j$. $W_{S,j,k}$ is the $k^{th}$ machine at stage $j$, $j \in \{1, \ldots, m\}$, $k \in \{1, \ldots, M_j\}$. $S_{i,j,k}$ is the start of processing time of job $i$ at stage $j$ on machine $k$. $C_{i,j,k}$ is the end of processing time of job $i$ at stage $j$ on machine $k$. $T_{w_{i,j,k}}$ is the processing time of job $i$ at stage $j$ on machine $k$. $n_{j,k}$ is the number of jobs allocated on machine $k$ at stage $j$.

B Model Variables and Constraints
- $A_{t_{i,j,k}}$ is a binary variable which is equal to 1 if job $i$ is assigned to machine $k$ at stage $j$ and is equal to 0 otherwise;
- $C_{i,j,k} = S_{i,j,k} + T_{w_{i,j,k}}$, $i \in \{1, 2, \ldots, n\}$, $j \in \{1, 2, \ldots, m\}$ (1)
- $C_{i,j,k} \leq S_{i+1,j,k}$, $i \in \{1, 2, \ldots, m-1\}$, $j \in \{1, 2, \ldots, m\}$, $k \in \{1, \ldots, M_j\}$ (2)
- $n_{j,k} = \sum_{i=1}^{n} A_{t_{i,j,k}}$ (3)

In the above constraints, constraint (1) describes the relationship between the start and the end of job $i$, no matter which stage it is; constraint (2) ensures that a job can’t be processed at next stage until it has being processed in the current one; constraint (3) shows that the total number of jobs on machine $k$ is equal to the number of all jobs which assigned to machine $k$.

III MATHMETICAL MODE OF HFS-LBSP

A. The HFS-LB Cost Based on Workstation Processing Time

$$T_{s_{j,k}} = \sum_{i=1}^{n} T_{w_{i,j,k}} \cdot A_{t_{i,j,k}}$$ (4)

In equation (4), $T_{s_{j,k}}$ is the sum of processing time on machine $k$ at stage $j$.

$$\bar{T}_{w_j} = \left( \frac{\sum_{i=1}^{n} \sum_{j=1}^{M_j} (T_{w_{i,j,k}} \cdot A_{t_{i,j,k}})}{M_j} \right)$$ (5)

In equation (5), $\bar{T}_{w_j}$ is the average value of the $M_j$ parallel machines at stage $j$.

$$T_{lb} = \sum_{j=1}^{M_j} \left( \frac{\sum_{k=1}^{n} (T_{s_{j,k}} - \bar{T}_{w_j})^2}{M_j} \right)$$ (6)

In equation (6), take the sum of the difference value between $T_{s_{j,k}}$ and $\bar{T}_{w_j}$ of all workstations as the workstation processing time load balancing cost which is the main evaluation index to measure the parallel machine load balancing degree.

B. Total Parallel Machine Processing Waiting Time

$$T_{m_{j,k}} = \left[ \max \{C_{i,j,k} - A_{t_{i,j,k}}\} - \min \{S_{i,j,k} - A_{t_{i,j,k}}\} \right] n_{j,k} \geq 2$$ \hspace{1cm} (7)

In equation (7), $n_{j,k} = \sum_{i=1}^{n} A_{t_{i,j,k}}$ is the job numbers allocated on parallel machine $j$. $T_{m_{j,k}}$ is the sum of waiting time between continuous jobs on machine $j$. $\max \{C_{i,j,k} - A_{t_{i,j,k}}\}$ is the max processing completion time of the jobs assigned to parallel machine $j$. $\min \{S_{i,j,k} - A_{t_{i,j,k}}\}$ is the earliest processing starting time of the jobs allocated on parallel machine $j$. The difference value of them is the time-span of manufacturing task on parallel machine $j$ and then subtract the effectiveness processing time $\sum_{j=1}^{M_j} (T_{s_{j,k}} \cdot A_{t_{i,j,k}})$ on machine $j$ can get the processing waiting time $T_{m_{j,k}}$ of machine $j$.

$$T_{wt} = \sum_{j=1}^{M_j} \left( \frac{M_j}{\sum_{k=1}^{n} T_{m_{j,k}}} \right)$$ (8)

In equation (8), $T_{wt}$ is the total waiting time of all parallel machines, which is the sum of waiting time of all parallel machines in processing, and then make it as the auxiliary evaluation index to measure the parallel machines’ load balancing degree.

C. Comprehensive Cost of HFS-LBSP

Before taking the weighted summation, normalization processing is need to be done to make the two evaluation index are within
one magnitude, which help to control the effect of the two index on HFS-LBSP more effectively, $f_{TLB}$ and $f_{WT}$ is the normalized value of $TL_b$ and $WT_t$.

$$f_{LB} = \alpha_1 \cdot f_{TLB} + \alpha_2 \cdot f_{WT}$$  \hspace{1cm} (11)

The weight value $\alpha_1$ and $\alpha_2$ are met the constraint: $\alpha_1 + \alpha_2 = 1$. The optimization objective is to minimize $f_{LB}$ of the HFS-LBSP, the smaller value of $f_{LB}$ the better optimization effect is.

### IV ALGORITHM DESIGN

DPSADE algorithm has made some improvements on standard DE algorithm, the key of it self-adaptive parameter adjusting strategy and cooperative evolution mechanism setting.

#### A. Encoding and Decoding

A real coding method based on matrix is proposed in this paper, using vector sequences instead of matrix to express chromosome, which is more visual and simple. The gene $a_{ij}$ of the population means job $i$ is operated on machine $a_{ij}$ at stage $j$, $\lfloor \cdot \rfloor$ means the floor of $a_{ij}$, $i \in [1,2,\ldots,n]$ and $j \in [1,2,\ldots,m]$, $a_{ij} = \text{rand}(1,M_j +1)$, each individual $X_{np} = \{a_{11},a_{12},\ldots,a_{ij},\ldots,a_{nm}\}$ shows the workstation allocation status of all jobs through HFS production processing.

#### B. Novel Evolution Mechanism

Before the evolutionary computation of each generation, classify the population into two parts based on fitness value, the individuals according with evolutionary trend is classified into elite population, otherwise to common population.

1) Respective evolution mode of the two population: DE/rand/1/bin\(^{[10]}\).

2) Cooperative evolution mode of the two population: DE/best/2/bin\(^{[10]}\), select two individual from the two Sub-population, then do a crossover operation to construct a new individual.

### C. Special Self-adaptive Parameter Adjusting Strategy

The cross operator $CR$ and mutation operator $F$ automatically change along with the evolution stop iteration, formula (20),(21) is the relationship between evolution parameter and stop iterations.

$$CR = CR \cdot \text{rand}(0,1) \cdot 2^{\frac{\text{StopGen}}{\text{StopGen}_{\text{max}}}} \pi$$ \hspace{1cm} (20)

$$F = F \cdot \text{rand}(0,1) \cdot 2^{\frac{\text{StopGen}}{\text{StopGen}_{\text{max}}}} \pi$$ \hspace{1cm} (21)

The two population have their own cross operators and mutation operators variation range, the cross operator and mutation operator adjustment range of common population is small, which contribute to keep the stability of excellent individual while the cross operator and mutation operator adjustment range of common population is big to strengthen the evolution energy.

### D. Procedure of The DPSADE Algorithm

Step 1: Initialize population, evolution parameters and iteration times value $\text{gen} = 0$.

Step 2: Classify initial population into elite population $\text{Pop1}$ and common $\text{Pop2}$, the total number of $\text{Pop1}$ is $N_1$ and $\text{Pop2}$ is $N_2$, $N = N_1 + N_2$.

Step 3: Conduct DE operation on $\text{Pop2}$, select three individuals to generate new individual $X_{new}$ by DE/rand/1/bin mode. Evaluate the fitness value of new individual $X_{new}$, if $X_{new}$ superior to any one of $\text{Pop1}$, then using $X_{new}$ to replace the worst individual of $\text{Pop1}$, otherwise, estimate $X_{new}$ whether superior to parent individual $X_i$, if it is, then using $X_{new}$ to replace $X_i$, otherwise, abandon it.

Step 4: Conduct DE operation on $\text{Pop2}$ select three individuals to generate new individual $X_{new}$ by DE/rand/1/bin mode. Evaluate the fitness value of new individual $X_{new}$, if $X_{new}$ is superior to any one of...
Step 5 Find out the best individual of Pop1, select two individuals from the two population respectively, then generate new individual $X_{new}^{*}$ by DE/best/2/bin mode. Evaluate the fitness value of new individual $X_{new}^{*}$, if $X_{new}^{*}$ superior to any one of Pop1, then using $X_{new}^{*}$ to replace the worst individual of Pop1, otherwise, abandon it.

Step 6 When all individual of the pop2 complete a evolution, if there are no individual were replaced in Pop1, then recording stop iteration StopGen, readjusting the cross operator $CR_1$, $CR_2$, $CR_3$, and mutation operator $F_1$, $F_2$, $F_3$.

Step 7 If the ending condition matched, then exit. Otherwise, jump to step 2.

V SIMULATION AND ANALYSIS

A. Evaluation Index Design

The load balancing cost $T_{lb}$, total processing waiting time of workstation $T_{wt}$, comprehensive cost $f_{LB}$, and $C_{max}$, were set as the evaluation index.

B. Simulation Results

In this section, a problem with 3 stages and 8 jobs is used to test the performance of the proposed algorithm and comparison. The number of machines in each stage is 3, 2, 2. The processing time of each job in every machine is a random number between [15, 30]. Each algorithm run 10 times, select the best solution to fill in Table 1.

| Scheme | Evaluation index |
|--------|------------------|
|        | $f_{LB}$ | $T_{lb}$ | $T_{wt}$ | $C_{max}$ | CPU |
| DE     | 0.0462   | 9.57    | 2       | 147       | 19.39 |
| SADE   | 0.0414   | 10.95   | 0       | 151       | 20.41 |
| DPSADE | 0.0331   | 8.74    | 0       | 152       | 26.78 |

It can be concluded from table 11, The HFS-LB comprehensive cost $f_{LB}$ of DPSADE algorithm is decreased by nearly 28.35% and 20.04% respectively compared to DE and SADE algorithm. Comparing the $C_{max}$ of the three group schemes, it is obvious that load balancing can make the completion time a little larger, but the largest deviation is no more than 3.40%, which is within the acceptable limits. The program execution time of DPSADE algorithm is extend by 38.11% compared with DE algorithm, which shows that in the program execution period, bi-population reconstruction and adaptive adjustment of parameters have cost parts program execution time.

![Fig. 1 The scheduling results’ gantt graph of scheme 3](image)

It can be seen from the figure 2, in the initial evolution stage, three schemes’ fitness drooped rapidly. DE have a weak evolution energe and got into local extremum after 110 generation. SADE and DPSADE algorithm can keep evolution energe during the evolution process, and obtain better solution. But Comparing the evolution curve of them, it is obvious that for
same evolution generation, the optimization
effect of DPSADE algorithm is always better.

VI CONCLUSION

This study has investigated the DPSADE
algorithm for the HFS load balancing scheduling
problem. The proposed DPSADE algorithm
involved a new bi-population structure and a
cooperative evolution Mechanism, by the flowing
of excellent individual from common population to
elite population to strengthen the evolution
advantage and keep evolution energy. Additionally,
a self-adaptive parameter adjusting strategy along
with stop iterations was designed to enhance the
ability to jump out of local extreme value.
Simulation result shows that when to solve HFS
load balancing scheduling problem, the DPSADE
algorithm can get better solution.

ACKNOWLEDGMENTS

This work was financially supported by the
Educational Commission of Liaoning Province
Science and Technology Research Projects
(No. L2013237). And the Nation S&T Major
Project, China.

Reference

[1] Thomopoulos N T. Mixed model line balancing
with smoothed station assignments[J]. Management
Science, 1970, 16(9): 593-603.

[2] Keskinurturk T, Yildirim M. B. An ant colony
optimization algorithm for load balancing in
parallel machines with sequence-dependent
setup times[J]. Computers and Operations
Research, 2012, 39(6): 1225-1235.

[3] ZHANG Chaoyong. Improved NSGA-II for the
Multi-objective Flexible Job-shop Scheduling
Problem. Journal of Mechanical Engineering, 2010,
46(11): 156-164.

[4] Rong-Hwa Huang. Flexible job shop scheduling
with due window-a two-pheromone ant colony
approach[J]. International Journal of Production
Economics, 2013, 141(2): 685-697.

[5] Kouki, Samia. A load balanced distributed
algorithm to solve the permutation flow shop
problem using the grid[C]. Proceedings-15th IEEE
International Conference on Computational
Science and Engineering, 2012: 146-153.

[6] Zhan Y, Qiu C. H. A Hybrid Genetic
Algorithm for Hybrid Flow Shop Scheduling
with Load Balancing[J]. Key Engineering
Materials, 2008, 392-394: 250-255.

[7] LIU Jianguo, ZHU Hengming. An immune
algorithm for load balancing of hybrid flow
shop scheduling[J]. Xi’an Dianzi Keji Daxue
Xuebao, 2006, 33(4): 655-659.

[8] Price K, Storn R. Differential evolution[J]. Dr
Dobbs Journal, 1997, 22(4): 18-23.

[9] WANG Wanliang, YAO Minghai. Hybrid
Flow-shop Scheduling Approach Based on Genetic
Algorithm[J]. Journal of system simulation, 2002,
14(7): 863-864.

[10] WU Lianghong, WANG Yaonan, ZHOU Shaowu,
et al. Research and application of pseudo parallel
differential evolution algorithm with dual
subpopulations[J]. Control Theory and Applications,
2007,24(3):453-458.