Modified Estimation of Distribution Algorithm for Solving Flow-shop Scheduling Problem with Setup Times

Mengxuan Feng
School of Automation
Nanjing University of Science and Technology
Nanjing, China
fmx0268@163.com

Jianshou Kong
School of Automation
Nanjing University of Science and Technology
Nanjing, China
kongjs77@163.com

Lingyan Liu
School of mechanical engineering
Nanjing University of Science and Technology
Nanjing, China
llylgy01@163.com

Guodong Liu
Automatic Control
Northern Institute of Automatic Control Technology
Taiyuan, China
liugd2003@163.com

Shanshan Zhao
School of Automation
Nanjing University of Science and Technology
Nanjing, China
Shans_zhao@163.com

Yue Zhang
School of Automation
Nanjing University of Science and Technology
Nanjing, China
zhang_y@njust.edu.cn

***Abstract*—To solve the flow-shop scheduling problem with sequence dependent setup times, the modified estimation of distribution algorithm was proposed. Considering the setup times dependent on sequence, multi-probability matrixes are employed in the probability model to denote the relative information about jobs in this problem. Meanwhile, the updating mechanism and an automatically adjusting method are improved to adapt to the new probability. A local search is designed to be able to jump out of the local optimum and improve the global optimization ability. Finally, simulation results are compared with other algorithms to demonstrate the effectiveness of the modified EDA.**

Keywords—flow-shop scheduling, setup times, the estimation of distribution algorithm, minimizing makespan.

I. INTRODUCTION

Setup times in general are either ignored or considered as a part of processing time by existing algorithms of the flow-shop scheduling problem. In actual production, it is necessary to carry out some operations such as replacing tools, setting machines and sweeping up. The time spent on these operations is independently called setup time, and should not be contained in processing time. As stated above, it is important to research the flow-shop scheduling problem with sequence dependent setup times (FSP/SDST) in theory and practice .

The most study the FSP/SDST by genetic algorithm. Ruiz and Maroto[1] surveyed the flow-shop with sequence dependent setup times and machine eligibility, proposing a hybrid genetic algorithm(HGA). Kaweegitbundit[2] presented a genetic algorithm embedding the NEH heuristic in order to study the FSP/SDST. Ciavota, Minella and Ruiz[3] presented an iterated pareto greedy algorithm with restarting mechanism minimizing the makespan for the FSP/SDST. Sioud and Gagne[4] proposed an enhanced migrating birds optimization(EMBO) based on the STH heuristic, and verified the effectiveness of the algorithm for the FSP/SDST.

Estimation of Distribution Algorithm(EDA) is a population evolution based on statistics theory[5]. The EDA employs a probability model for good characteristics of the population. By sampling the probability model to produce new solutions, the EDA insures a direction of evolution. Currently, EDA has been adopted to obtain various shop scheduling solutions. Jarboui, Eddaly and Siarry[6] proposed an EDA added a variable neighborhood search to obtain the flow-shop scheduling solution. Wang et al.[7] surveyed the flexible flow-shop scheduling problem, proposing an effective EDA. While sampling to generate new solutions, it can be hard to continue to sample with the sum of the probability model getting smaller. Tang et al.[8] surveyed the flexible flow-shop scheduling problem, designing an automatically adjusting method for the probability model to solve this problem in sampling.

Based on these analyses, this paper put forward a modified estimation of distribution algorithm (MEDA) to study the flow-shop scheduling problem with sequence dependent setup times with makespan criterion (FSP/SDST-Cmax). Sequence dependent setup times mean, for each job, the setup times depend not only on the machines but also on the job right before. Therefore, the probability model is designed into multi-probability matrixes to replace a single probability matrix. Accordingly, the updating mechanism and the automatically adjusting method are improved to adapt to the new probability model. In this condition, a roulette wheel employs conditional probability instead of total probability in sampling to produce new solutions. Besides, a local search is designed to be able to jump out of the local optimum with some probability. Finally, simulation results are compared with other algorithms to demonstrate the effectiveness of the MEDA in the FSP/SDST-Cmax.

II. FLOW-SHOP SCHEDULING MODEL WITH SETUP TIMES

A. Problem Description

FSP/SDST-Cmax can be described as follows: There are n jobs, which are in a specific sequence, to be processed by m machines. It is necessary to consider a setup time before continuing the next job. This setup time is not only related to the machines but also to the job sequence. The objective is to determine the job processing sequences to minimize the

This paper is supported by the National Science Foundation for Young Scientists of China (Grant NO.51705256).

maximum completion time of all jobs. Meanwhile, it assumes beforehand: In the very initial moment, jobs and machines are unoccupied; There is an infinite buffer between any two machines; Any operation is uninterruptible; The job sequence is the same on each machine; Each machine can only process jobs one after another. Similarly, a job cannot be on multiple machines at one time.

B. Variable Definitions

i : the job number.

k : the machine number.

n : the total number of jobs.

m : the total number of machines.

π : one sequence of jobs.

l : the sequence position number of jobs.

$\pi(l)$: job with position number l .

$S_{\pi(l),k}$: starting time of job $\pi(l)$ on k machine.

$P_{\pi(l),k}$: processing time of job $\pi(l)$ on k machine.

$C_{\pi(l),k}$: completion time of job $\pi(l)$ on k machine.

$St_{\pi(l-1)\pi(l),k}$: setup time of job $\pi(l)$ on k machine right after job $\pi(l-1)$.

$C_{\max}(\pi)$: the maximum completion time of the job sequence π .

C. Mathematical Model

The optimization objective is minimizing makespan. The mathematical model can be formulated:

$$Makespan = C_{\max}(\pi) \tag{1}$$

Minimize makespan:

$$\min(C_{\max}(\pi)) = \min(\max(C_{\pi(l),m})) \tag{2}$$

Subject to:

$$C_{\pi(l),\mathrm{k}} = S_{\pi(l),k} + P_{\pi(l),k} \tag{3}$$

$$S_{\pi(1),1} = 0, l = 1, k = 1 \tag{4}$$

$$S_{\pi(1),k} = C_{\pi(1),k-1}, l = 1, k = 2,3,...,m \tag{5}$$

$$S_{\pi(l),1} = C_{\pi(l-1),1} + St_{\pi(l-1)\pi(l),1},$$

$$k = 1, l = 2,3,...,n \tag{6}$$

$$S_{\pi(l),k} = \max(C_{\pi(l-1),k} + St_{\pi(l-1)\pi(l),k}, C_{\pi(l),k-1}),$$

$$l = 2,3,...,n, k = 2,3,...,m \tag{7}$$

$$\max(C_{\pi(l),m}) = C_{\pi(n),m} \tag{8}$$

Equation (3) defines the completion time of job $\pi(l)$ on k machine; Equation (4) (5) define the starting time of job $\pi(1)$ on machine 1 and machine k, and ensure a job cannot be on multiple machines at one time; Equation (6) (7) define the starting time of job $\pi(l)$ on machine 1 and machine k, and indicate each machine can only process jobs one after another; Equation (8) indicates that makespan is equal to the completion time of job $\pi(n)$ on machine m.

III. The Modified Estimation of Distribution Algorithm

This section presents a MEDA to solve the FSP/SDST-Cmax, and explains the solution representation, the population initialization, the probability model, the updating mechanism and the automatically adjusting method, and the probabilistic-jumping local search.

A. Brief Introduction of the basic EDA

EDA is a population evolution based on statistics theory. The EDA updates the probability model from promising sub-population, then the model is sampled to produce new solutions. It models the distribution of promising sub-population to avoid destroying good patterns like the traditional evolution algorithm does. The basic working principle of EDA illustrates as follows:

Step 1: Initialize a population contains N solutions.

Step 2: Sort the solutions by fitness values in descending order.

Step 3: Select the top E solutions. Get the statistics about the positions in these solutions, and update the probability matrixes.

Step 4: Sample the updated probability matrixes, and generate new solutions.

Step 5: Arrange the new solutions in descending order of fitness values.

Step 6: Select the top E solutions for the local search.

Step 7: If the termination condition is not met, return to step 2; Otherwise, output optimal solution.

B. Solution Representation and Initialization Mechanism

The population is composed by the solutions of the FSP/SDST-Cmax. Equation (9) illustrates that each solution can be constructed by an order of n jobs. The order indicates the sequence of processing jobs on each machine.

$$\pi = \{\pi(1), \pi(2), ..., \pi(n)\} \tag{9}$$

NN-MNEH heuristic algorithm and random generation method are used for initialization. The NN-MNEH heuristic[9] obtains a part of the initial population to guarantee the quality of solutions. And the random generation method generates the rest randomly to ensure the diversity of population.

C. Probability Model

It is critical to construct an appropriate probability model for the EDA. The probability model represents the characteristics of distribution and affects the sampling result. Considering sequence dependent setup times, the probability model employs n probability matrixes. Each probability matrix reflects a specific position and contains relative information about the jobs in this position and the jobs in the former position. Therefore, a probability model is designed as n probability matrixes as follows:

(1) Job $\pi(1)$ without the previous job:

$$P_1(g) = [\rho_1(g), \rho_2(g), ..., \rho_n(g)] \quad (10)$$

where $P_1(g)$ symbolizes a probability matrix which is at the first position in the $g\text{-}th$ generation. Besides, in the equation $\rho_i(g) = p(\pi(1) = i)$, $\rho_i(g)$ represents the probability that the job i is selected at the first position. The initial probability of $\rho_i(g)$ is $\rho_i(0) = 1/n$. It represents each job has the equal probability to be selected at the first position and the sum of probabilities is 1 in the initial.

(2) Job $\pi(l)$ with the previous job $\pi(l-1)$:

$$P_l(g) = \begin{bmatrix} 0 & \cdots & \cdots & g) \\ \vdots & \ddots & \vdots & \\ \rho_{n1l}(g) & \cdots & & \end{bmatrix} \quad \leq n \quad (11)$$

where $P_l(g)$ symbolizes a probability matrix which is at the $l\text{-}th$ position in the gth generation. Besides, in the equation $\rho_{ijl}(g) = p(\pi(l-1) = j, \pi(l) = i)$, $\rho_{ijl}(g)$ represents the probability that the job i is selected at the $l\text{-}th$ position and right after job j. The initial probability of $\rho_{ijl}(\mathrm{g})$ is $\rho_{ijl}(0) = \frac{1}{n(n-1)}$. It represents each job has the equal probability to be selected at the $l\text{-}th$ position and the sum of probabilities is 1 in the initial.

D. Updating Mechanism

The elite solutions, which are ranked in the front, are employed to update the probability model in each generation. The updating mechanism is critical to the direction of evolution. To adapt to the new probability model, the updating mechanism is improved accordingly as follows:

(1) Updating $P_1(g)$:

$$\rho_i(g+1) = (1-\alpha)\rho_i(g) + n_i(g) \times \frac{\alpha}{E} \quad (12)$$

where α represents a learning rate, E represents the number of elite solutions, $n_i(g)$ represents the count of the case that job i is at the first position in elite solutions.

(2) Updating $\rho_{ijl}(g)$:

$$\rho_{ijl}(g+1) = (1-\alpha)\rho_{ijl}(g) + n_{ijl}(g) \times \frac{\alpha}{E} \quad (13)$$

where α represents a learning rate, E represents the number of elite solutions, $n_{ijl}(g)$ represents the count of the case that job i is at the $l\text{-}th$ position and right after job j at the time.

E. Generating Solutions

The MEDA employs a roulette wheel to sample the updated probability matrixes. The roulette wheel is designed to adapt to the probability model. While one job is determined in a position, it is necessary to adjust the probability matrixes. Otherwise, the sum of the probabilities will reduce and be hard to continue sampling. The way to generate new solutions is conducted as follows:

Step 1: Sampling the probability matrixes by the roulette wheel.

(1) Sampling to select job $\pi(1)$:

In fact, $P_1(g)$ is a $1 \times n$ dimensional matrix, just employing the traditional roulette wheel to sample.

(2) Sampling to select job $\pi(l)$:

$\rho_{ijl}(g)$ is an $n \times n$ dimensional matrix, the roulette wheel is designed as follows:

$\rho_{ijl}(g)$ indicates the probability of job i is at the lth position and right after job j. Therefore, it can be represented as the joint probability $p(\pi(l-1) = j, \pi(l) = i)$.

According to Bayes theorem:

$$p(\pi(l) = i \mid \pi(l-1) = j) = \frac{p(\pi(l-1) = j, \pi(l) = i)}{p(\pi(l-1) = j)} \quad (14)$$

$$p(\pi(l-1) = j) = \sum_i p(\pi(l-1) = j, \pi(l) = i) \quad (15)$$

Transform equation (14) (15):

$$p(\pi(l-1) = j) = \sum_i \rho_{ijl}(g) \quad (16)$$

$$p(\pi(l) = i \mid \pi(l-1) = j) = \frac{\rho_{ijl}(g)}{\sum_i \rho_{ijl}(g)} \quad (17)$$

Except using the conditional probability ($p(\pi(l) = i \mid \pi(l-1) = j)$) in the equation (17), other operations are same to the traditional roulette wheel, which normally uses a total probability.

Step 2: Automatically adjusting the probability matrixes.

To ensure the sum of the probabilities of the rest jobs is 1, the $n - l$ matrixes are adjusted as follows:

$$\rho'_{srt}(g) = \frac{\rho_{srt(g)} \times p(\pi(t-1) = r)}{p(\pi(t-1) = r) - \rho_{irt}(g)},$$

$$s \neq i, r \in U, s \in U \tag{18}$$

where job s is the rest unselected, job i is selected by step 1, $\rho_{srt(g)}$ represents the probability that job s is selected at the t-th position and right after job r, U is the set of the rest jobs.

Meanwhile, in order to avoid job i reselected, the columns of job i are set to 0 in the $n-l$ matrixes as follows:

$$\rho'_{irt}(g) = 0, r \in U \tag{19}$$

Equation (18)(19) launch $\sum_s \sum_r \rho'_{srt}(g) = 1$. While sampling to select jobs, it ensures the sum of the probabilities is always 1 by adjusting the probability matrixes.

F. Probabilistic-jumping Local Search

The local search is able to jump out of the local optimum by some probability to accept an inferior solution. During the initial search, the convergence rate is fast with a small probability to accept that. And then the probability is increasing gradually when search for neighborhood solutions. Whereas, it is able to jump out of the local optimum by accepting the inferior solution with a high probability at the end of the search. The specific method is constructed as follows:

Step 1: Parameters initialization. System state $T = T_0$, initial system solution $\pi = \pi_0$, initial system target value $c = c_0$, system parameter $\varepsilon > 1$, number of outer iterations $iter = 0$, and neighborhood length $L = len$.

Step 2: If $iter < iter_{\max}$ and $T < T_{end}$, number of inner iterations $it = 0$; Otherwise, jump to step 3.

Step 2.1: If $it < L$, continue; Otherwise, $iter = iter + 1$, $T = \varepsilon T$ and jump to step 2.

Step 2.2: Perform the search operations to get a new solution π_1 and a target value c_1. Calculate $\Delta = c_1 - c$. If $\Delta \leq 0$, continue; Otherwise, jump to step 2.4.

Step 2.3: Accept the new solution $\pi = \pi_1$, $c = c_1$. $it = it + 1$, jump to step 2.1.

Step 2.4: Calculate $r = \exp(-\frac{\Delta}{T})$. Generate ξ randomly in $[0,1)$. If $r > \xi$, accept the new solution $\pi = \pi_1$, $c = c_1$. $it = it + 1$, and jump to step 2.1.

Step 3: Output $\pi_{\min}$, whose target value is $c_{\min}$ in the search history.

Moreover, Step 2.2 employs 3 search operations to generate new solutions. They are swap, inset and reverse.

(1) Swap:

Select two jobs in the job sequence π randomly and swap.

(2) Insert:

Select two jobs, $\pi(u)$ and $\pi(v)$ $(1 \leq u < v \leq n)$, in the job sequence π randomly. Insert $\pi(v)$ in front of $\pi^q(u)$.

(3) Reverse:

Select a block of the job sequence π randomly $(\text{block} = \{\pi^q(u), ..., \pi^q(v)\}, 1 \leq u < v \leq n)$. Reverse the job sequence of the block.

G. The Flow of the MEDA

The flow chart of the MEDA is illustrated in Figure 1.

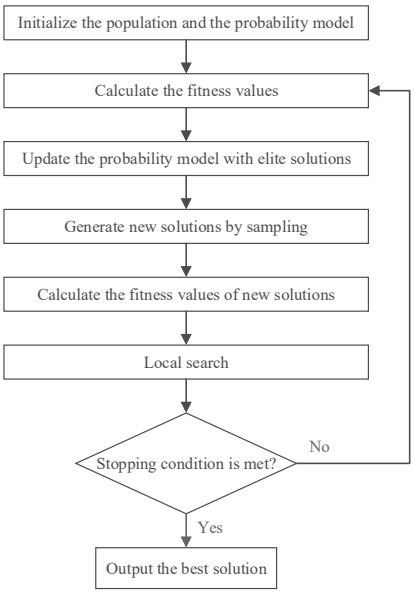

Fig. 1. The flow chart of the MEDA.

IV. Simulation Examples Experiment

To verify the effectiveness of the MEDA, the test set consists of 100 instances improved the benchmarks of Taillard[10], where $n = \{20, 50, 100\}$ and $m = \{5, 10, 20\}$. The processing times are integers evenly distributed in [1,99], and the setup times are integers evenly distributed in [0,10].

All algorithms are coded in C++. Independent 20 times simulations must be performed to run for each algorithm with 50 iterations.

A. Parameters Setting

The MEDA has 4 key parameters: population size N, learning rate α, parameter of the number of elite solutions δ and neighborhood length L, where the number of elite solutions is represented as $E = \delta \times N$.

To analyze the impact of the four parameters on the MEDA, Design of Experiment (DOE) is performed on the SDST10_ta001 instance adopting four factor levels as Table I.

TABLE I. FACTOR LEVEL

Parameters	Factor level			
	1	2	3	4
N	30	60	90	120
α	0.06	0.08	0.1	0.12
δ	0.1	0.2	0.3	0.4
L	3	6	12	18

Choosing the orthogonal array $L_{16}(4^4)$, the MEDA runs 20 times independently for every parameters combination. The orthogonal array and the response values are demonstrated in Table II .

TABLE II. RESPONSE VALUES

Number	Factor				Response
	N	α	δ	L	Values
1	1	1	1	1	1344.00
2	1	2	2	2	1337.40
3	1	3	3	3	1334.60
4	1	4	4	4	1338.90
5	2	1	2	4	1331.90
6	2	2	3	1	1335.10
7	2	3	4	2	1333.40
8	2	4	1	3	1334.40
9	3	1	3	4	1331.20
10	3	2	4	3	1331.40
11	3	3	1	1	1333.50
12	3	4	2	2	1330.90
13	4	1	4	4	1334.80
14	4	2	1	3	1332.80
15	4	3	2	2	1333.20
16	4	4	3	1	1334.30

From Table III , the population size is on the first rank, therefore it is the most important effect, while the neighborhood length has less effect than it. The last rank is the learning rate with the least effect. Besides, the performance of the MEDA is better, when the population size is level 3, the learning rate is level 3, the parameter of the number of elite solutions is level 2, and the neighborhood length is level 3.

Accordingly, the parameters are set as follows: $N = 90$, $\alpha = 0.1$, $\delta = 0.2$, $L = 12$.

TABLE III. RESPONSE TABLE

Level	N	α	δ	L
1	1338.73	1335.48	1336.18	1336.73
2	1333.70	1334.18	1333.35	1333.73
3	1331.75	1333.68	1333.80	1333.30
4	1333.78	1334.63	1334.63	1334.20
Delta	6.98	1.80	2.83	3.43
Rank	1	4	3	2

B. Results Comparing

To verify the performance of the MEDA, it is compared with EDA-IG[11], EMBO-STH[4] and HGA[1] in the FSP/SDST. The computational results are scored by the average of relative percentage deviation (ARPD) as follows:

$$RPD = \frac{ME - Best}{Best} \times 100\% \tag{20}$$

$$APRD = \sum RPD / num \tag{21}$$

Equation (20) defines relative percentage deviation (RPD), where ME is the result of the algorithm, Best is the optimum already known. Equation (21) defines APRD, where num is the size of a set of instances.

From Table IV , the MEDA has the best average of ARPD among all the algorithms. Comparing the MEDA with EDA-IG, it is noticed that the EDA is improved except the instances of 100×5 . It can be explained that it is the MEDA which employs the new probability model and automatically adjusting method, and replaces the IG operation to the probabilistic-jumping local search which improve the global search.

Among the MEDA, EMBO-STH and HGA, the MEDA performs the better effective than the others. EMBO-STH provides a better result than HGA, but the difference between EMBO-STH and the MEDA is noticeable. Meanwhile, it is indicated that the MEDA is robustness with a smaller standard deviation.

TABLE IV. COMPARISON OF MEDA, EDA-IG, EMBO-STH AND HGA

Instances	MEDA	EDA-IG	EMBO-STH	HGA
20×5	**0.27192**	0.66556	0.70867	2.41773
20×10	**0.53629**	0.98066	1.05348	2.89361
20×20	**0.66743**	1.0412	1.07931	2.57025
50×5	**0.81092**	1.29034	2.14189	2.9338
50×10	**1.85154**	2.57006	4.0809	5.19514
50×20	**2.14283**	2.82645	4.46579	4.87441
100×5	0.97941	**0.81869**	1.60688	2.74319
100×10	**1.22524**	1.4881	3.28086	5.11726
100×20	**1.97264**	2.26663	4.84342	4.34972
Average	1.16202	1.54974	2.58458	3.67723
STD	0.67856	0.80271	1.60648	1.17867

V. CONCLUSION

In this paper, a modified estimation of distribution algorithm (MEDA) is constructed to solve the flow-shop scheduling problem with sequence dependent setup times (FSP/SDST). Considering the setup times relied on sequence, multi-probability matrixes are employed in the probability model to denote the relative information about jobs in this problem. In order to adapt to the new probability model, the updating mechanism is improved. Meanwhile, a roulette wheel and an automatically adjusting method are designed to ensure sampling successfully. Besides, a probabilistic-jumping local search is designed to enhance the global search by a probability to help the local search jump out of the local optimum. Comparing the MEDA with three other algorithms, it verifies the MEDA is effective to solve the FSP/SDST, and has the better performance than other algorithms. It is significance to conduct to plan the practical shop scheduling.

REFERENCES

[1] Ruiz R, Maroto C. A genetic algorithm for hybrid flow shops with sequence dependent setup times and machine eligibility. European Journal of Operational Research, vol 169, pp. 781–800, 2006.

[2] Kaweegitbundit P. Comparison of heuristic for flow shop scheduling problems with sequence dependent setup times. Advanced Materials Research,vol 399, pp: 332-335, 2011.

[3] Ciavotta M, Minella G, Ruiz R: Multi-objective sequence dependent setup times permutation flow shop: A new algorithm and a comprehensive study. European Journal of Operational Research, vol 227, pp: 301-313, 2013.

[4] Sioud A, Gagne C. Enhanced migrating birds optimization algorithm for the permutation flow shop problem with sequence dependent setup

times. European Journal of Operation Research, vol 264,pp: 66-73, 2018.

[5] Wang S Y, Wang L, Fang C, XU Y. Advances inestimation of distribution algorithms. Control and Decision, vol 27, pp: 962-965, 2012.

[6] Jarboui B, Eddaly M, Siarry P. An estimation of distribution algorithm for minimizing the total flowtime in permutation flow shop scheduling problems[J]. Computers & Operations Research, vol 36, pp:2638–2646, 2009.

[7] Wang S Y, Wang L, XU Y, Zhou G. EDA algorithm for solving hybrid flow-shop scheduling problem. Acta automatica Sinica, vol 38, pp: 437-443, 2012.

[8] Wang F, Tang Q H, Rao Y Q , Zhang Y C, Zhang L P. Effcient estimation of distribution for flexible hybrid flow shop scheduling. Acta automatica Sinica, vol 43, pp: 280-293, 2017.

[9] Fuqing Z, Huan L, Yi Z. A discrete water wave optimization algorithm for no-wait flow shop scheduling problem. Expert Systems With Applications, vol 91, pp: 347–363, 2017.

[10] Ciavotta M, Minella G, Ruiz R. Multi-objective sequence dependent setup times permutation flowshop: a new algorithm and a comprehensive study. European Journal of Operational Research, vol 227, pp: 301–313, 2013.

[11] Ling W, Shengyao W, Xiaolong Z. A hybrid estimation of distribution algorithm for unrelated parallel machine scheduling with sequence-dependent Setup Times. IEEE/CAA Journal of Automatica Sinica, vol 3, pp: 235-246, 2016.