F O U N D A T I O N S O F C O M P U T I N G A N D D E C I S I O N S C I E N C E S
Vol. 49 (2024) No. 1
ISSN 0867-6356
DOI: 10.2478/fcds-2024-0001 e-ISSN 2300-3405

Multi-criteria Scheduling in Parallel Environment with Learning Effect

Xinbo Liu *, Yue Feng †, Ning Ding ‡, Rui Li, Xin Chen §

Abstract. This paper is devoted to the study of a multi-criteria scheduling problem on unrelated processors with machines' learning effect, with the goal of minimizing makespan, machine cost and maximal flow-time simultaneously, which is an NP-hard problem. An improved particle swarm optimization algorithm equipped with the overloaded operators, as well as a procedure of Levy flight, is proposed to generate the Pareto-optimal solutions. The experimental results show that the Levy flight strategy can effectively improve the performance of the algorithm, which can generate more non-dominated solutions, and slightly reduce the execution time of the process.

Keywords: multi-criteria, parallel processors scheduling, learning effect, particle swarm optimization, Levy flight

1. Introduction

Learning effect [6, 10] is one of the factors that are commonly considered in the scheduling field, especially in the problems related to human resources scheduling [20, 49]. For example, in scenarios such as doctor diagnosis, business consultation, software development or manual manufacture etc., the execution speed of the operator increases along with familiarity of the work, so that the actual processing time of the work is reduced. If we treat the operators working in parallel as several parallel machines, and the tasks assigned to them as jobs, all the above scenarios can be formulated as parallel-machine scheduling problems with learning effect [35, 52].

*SolBridge International School of Business, Woosong University, Daejeon, South Korea.
†School of Information Engineering, Liaoning Institute of Science and Engineering, Jinzhou, China.
‡College of Physical and Health, Dalian University of Technology, Panjin, China.
§School of Electronics and Information Engineering, Liaoning University of Technology, Jinzhou, China. Corresponding author: Xin Chen (chenxin.lut@hotmail.com)

In practical situations, the decision-makers always need to consider several criteria at the same time [32, 44]. In the scheduling domain, minimizing the makespan [3, 17] is definitely one of the key objectives for optimization (from the global point of the view). Besides, in the problems related to human resources scheduling, minimizing the total human cost (or we can say, machine cost) is also an important criterion [14, 29]. Moreover, along with the increment of the attentions on customer satisfaction, the duration time on solving customers' needs should not be too long, i.e., the flow-time of each job (from being released to completed) should be minimized [2, 39]. However, the above-mentioned goals are often contradictory, that is, the decision-makers have to make a trade-off among these criteria, and choose one solution from several reasonable ones.

In this paper, we consider a multi-criteria scheduling problem on parallel unrelated processors with machines' learning effect, setting the goal as minimizing makespan, machine cost and maximal flow-time simultaneously. Due to the NP-hardness [15] of the considered problem, we propose an improved particle swarm optimization (PSO) algorithm [40] to generate Pareto-optimal solutions [18]. Specifically, after giving the representation of the solutions, we overload the operators, i.e. $-$, $\times$ and $+$ in our PSO, and add a Levy flight procedure to avoid local optima. Finally, we verify the performance of our improved PSO algorithm through the computational experiments.

The rest of this paper is organized as follows. Some related work and the problem formulation are introduced in Section 2. Section 3 is devoted to the analysis and design of the improved PSO algorithm, which is verified by numerical experiments in Section 4. Finally, the whole paper is concluded and some future work is discussed in Section 5.

2. Preliminaries

2.1. Related work

Scheduling with multi-criteria is a very popular topic in the field due to actual productions, and fruitful results have been devoted to it. Choobineh et al. [13] considered a single machine scheduling problem with m-objective, and proposed a tabu search algorithm for it. They also used the numerical examples to study the behavior of tabu search, comparing the optimal solutions when $m = 3$. Yagmahan and Yenisey [50] studied a flow-shop scheduling problem with multi-objectives of makespan, total flow-time and total machine idle time, and designed an ant colony optimization algorithm to solve the problem. Wang et al. [47] presented a genetic algorithm based on immune and entropy principle to solve a multi-objective flexible job-shop scheduling problem, in which a fitness scheme based on Pareto-optimality was applied, and the immune and entropy principle was used to keep the diversity of individuals and overcome the problem of premature convergence. Bandyopadhyay and Bhattacharya [8] proposed a non-dominated sorting genetic algorithm-II for a parallel machine scheduling problem with three objectives, which are minimization of total cost due tardiness, minimiza-

tion of the deterioration cost and minimization of makespan, respectively. Ozdagoglu et al. [38] developed a dynamic priority approach for a group-based single machine scheduling problem, using multi-criteria decision making technique regarding setup time and machine utilization. With the aim of designing an energy-efficient scheduling in a flexible job-shop system, Mokhtari and Hasani [33] developed a multi-objective optimization model to minimize total completion time, total energy cost and maximize the total availability of the system, and then gave an enhanced evolutionary algorithm for it. Recently, Abedi et al. [1] focused on an energy-efficient job-shop scheduling problem within a machine speed scaling framework, where productivity was affected by deterioration. They proposed a multi-population, multi-objective memetic algorithm, in which some local search methods were designed based on a new disjunctive graph introduced to cover the solution space. Saber and Ranjbar [42] considered a permutation flow-shop scheduling problem with the goal of minimizing total tardiness and total carbon emissions. After presenting a mixed integer programming model, they developed a multi-objective decomposition-based heuristic algorithm, as well as a multi-objective variable neighborhood search algorithm, respectively. Lei et al. [26] investigated a multi-objective distributed unrelated machine scheduling problem, and adopted an improved artificial bee colony algorithm to minimize makespan and total tardiness simultaneously. Caldeira and Gnanavelbabu [11] utilized a multi-objective discrete Jaya algorithm to find Pareto-optimal solutions for a flexible job-shop scheduling problem, in which the minimization of makespan, total workload of machines, as well as workload of critical machine were considered simultaneously. Since multi-criteria scheduling has always been one of the hot spots in scheduling field, there are a large number of survey articles for reference, e.g. [21, 25, 45].

The phenomenon of learning effect was discovered in 1936 [48], and introduced into the scheduling field from 1999 [9], when a single machine scheduling problem with this fact was considered. In this seminal paper, Biskup [9] proposed that the actual time needed for processing one job decreases along with the number of repetitions of the similar jobs. Therefore, the actual processing time of job J_j can be represented as

$$p_j^r = p_j \times r^{\alpha} \tag{1}$$

if J_j is scheduled on position r on one machine, in which p_j is the original processing time of J_j and $\alpha < 0$ is a learning factor. Then, Lee and Wu [24] extended this concept to a two-machine flow-shop system, with the setting that $p_{ij}^r = p_{ij} \times r^{\alpha}$, where p_{ij} is the original processing time of job J_j on machine M_i. Mosheiov and Sidney [36] suggested that the learning factor should not be identical, but related to specific jobs. So they denoted the specific learning factor of J_j as α_j, and improved Equation (1) to $p_j^r = p_j \times r^{\alpha_j}$. Vahedi Nouri et al. [46] considered a flow-shop scheduling problem with learning effect and flexible maintenance activities, and proposed a hybrid firefly-simulated annealing algorithm to solve it. Yeh et al. [51] discussed a parallel machine scheduling problem with fuzzy processing time and learning effect, and designed two meta-heuristic algorithms (simulated annealing and genetic algorithm) for it. Bai et al. [7] modeled the real-world workshop assembly process into a flow-shop scheduling problem with release date and learning effect, and then solved it by a mixed integer

programming model, a branch-and-bound algorithm and a class of shortest processing time available (SPTA)-based heuristics, respectively. Przybylski [41] introduced a new model of parallel machine scheduling with job processing time described by proper Riemann integrals of a given function, and showed that the problems with integral-based learning effect can be solved in polynomial time. Li et al. [27] investigated a parallel machine scheduling problem with position-dependent deteriorating jobs and DeJong's learning effects in an uncertain manufacturing system, and solved it by a fuzzy simplified swarm optimization algorithm with a local search strategy. To have a comprehensive investigation on the topic of scheduling with learning effect, we recommend some surveys, e.g. [5, 10].

2.2. Problem definition

In this paper, we consider a parallel unrelated machines scheduling problem with learning effect and multi-criteria, which are the minimization of makespan, machine cost and maximal flow-time, respectively. More formally, the problem can be described as follows.

Input: there are m parallel unrelated machines and n jobs, and each job should be processed by one of the machines, in which the original processing time of job J_j on machine M_i is p_{ij} ($1 \leq i \leq m$ and $1 \leq j \leq n$). Each job has a release time r_j, and each machine has a unit operation cost u_i. Machines have the abilities of learning, so that if J_j is scheduled on the r-th position of M_i, its actual execution time is $p_{ij}^r = p_{ij} \times r^{\alpha}$ [9, 24], where $\alpha < 0$ is the learning factor.

Output: schedule the n jobs on the m machines without preemption, to find the Pareto-optimal solutions with the criteria of minimizing makespan (C_{max}), machine cost (U_{cost}) and maximal flow-time (L_{max}), in which

$$C_{max} = \max_{1 \leq j \leq n} \{C_j\}, \tag{2}$$

$$U_{cost} = \sum_{i=1}^{m} u_i \times \Big(\sum_{J_j \in M_i} p_{ij}^r \Big), \tag{3}$$

$$L_{max} = \max_{1 \leq j \leq n} \{C_j - r_j\}, \tag{4}$$

and C_j is the completion time of J_j.

Based on the classical three-field notation [16] in the scheduling field, the problem considered in this paper can be presented as $Rm|r_j,\ LE|^{\#}(C_{max}, U_{cost}, L_{max})$, where LE in β field stands for learning effect, and $^{\#}(C_{max}, U_{cost}, L_{max})$ in γ field means that this is a trade-off optimization, i.e., to find the Pareto-optimal solutions based on these three criteria.

3. Particle swarm optimization with Levy flight

Problem $Rm|r_j,\ LE|^{\#}(C_{max}, U_{cost}, L_{max})$ is NP-hard since the classical parallel machines scheduling is already NP-hard [15]. Therefore, to solve the problem effectively, we propose an improved Particle Swarm Optimization algorithm with a strategy of Levy flight ($LFPSO$ for short), to find the Pareto-optimal solutions.

3.1. Basic concept of PSO

Motivated from the foraging process of birds or fishes, PSO [23] is a kind of population-based meta-heuristic algorithms, which is widely used for solving the combinatorial optimization problems such as scheduling [43], bin packing [28] or routing [31], etc.

In the general framework of a PSO, the whole population contains N independent particles. Each particle P_k ($1 \leq k \leq N$) has a position x_k and a velocity v_k, in which x_k represents a feasible solution for the considered problem, while v_k expresses an update trend of P_k in a certain sense. The searching process of a PSO is to use the interactions among these particles, that is, the "best-found" position of one particle will affect the movements of the others, so that the whole population has a move trend towards the optima. More precisely, in the $(t+1)$-th iteration, the new velocity (v_k^{t+1}) and position (x_k^{t+1}) of particle P_k is influenced by the "best" position already found by itself ($pBest_k^t$), and the one found by the whole population ($gBest^t$), with the following formulas.

$$v_k^{t+1} = wv_k^t + c_1R_1 \times (pBest_k^t - x_k^t) + c_2R_2 \times (gBest^t - x_k^t) \tag{5}$$

$$x_k^{t+1} = x_k^t + v_k^{t+1} \tag{6}$$

In Equation (5), w is the weight to reflect the influence by the previous velocity of P_k, while c_1 and c_2 are the ones reflecting the influences from the local best-found solution ($pBest_k^t$) and the global best-found solution ($gBest^t$). Moreover, R_1 and R_2 are the random variables to keep the variety of the searching process. When utilizing a PSO to solve a specific well-defined problem, such as $Rm|r_j,\ LE|^{\#}(C_{max}, U_{cost}, L_{max})$ in this paper, we need to design the encoding for x_k, v_k and other parameters, and overload the operators of $-$, $\times$ and $+$ in Equations (5) and (6). The details will be given in Section 3.3.

3.2. Strategy of Levy flight

The strategy of Levy flight (LF for short) was proposed by Levy [37], who found that some marine animals such as tuna, swordfish, and sharks sometimes follow strict mathematical strategies when foraging, instead of disordered movements. Such mathematical principle related to chaos theory can be widely used in random measurement or computation, to improve the effectiveness of the simulations [37].

Applying LF strategy to a searching algorithm, one can extend the exploration space through occasional "long jump", to avoid its falling into the local optima, and improve the variety and flexibility of the solutions [4]. Specifically for a PSO algorithm (e.g. [19, 22]), after Equation (6), we can update the position of particle P_k with a certain probability of Levy flight, i.e.,

$$x_k^{t+1} = x_k^{t+1} + \delta \oplus Levy(\lambda), \tag{7}$$

in which δ is a random step-length, $\oplus$ means the scalar product operation, and $Levy(\lambda)$ is a probability function generated from Mantegna algorithm [30], by the following process.

$$Levy(\lambda) \sim \frac{u}{|v|^{\frac{1}{\lambda}}} \tag{8}$$

$$u \sim N(0, \sigma_u^2) \tag{9}$$

$$v \sim N(0, 1) \tag{10}$$

$$\sigma_u = \left\{ \frac{\Gamma(1+\lambda) sin(\frac{\pi\lambda}{2})}{\Gamma\left[\frac{1+\lambda}{2}\right] \lambda 2^{\frac{\lambda-1}{2}}} \right\}^{\frac{1}{\lambda}} \tag{11}$$

In Equation (11), $\Gamma()$ is the standard Gamma function, and we set $\lambda \in [0.30, 1.99]$ in this paper.

3.3. $LFPSO$ for $Rm|r_j,\ LE|^{\#}(C_{max}, U_{cost}, L_{max})$

We give our $LFPSO$ algorithm for problem $Rm|r_j,\ LE|\ ^{\#}(C_{max}, U_{cost}, L_{max})$ in this part. We first introduce the solution representation, i.e., the encoding of the parameters in Equations (5)-(7), and then overload the operators such as $-$, $\times$ and $+$. After setting the initial population and terminal conditions, the whole process of $LFPSO$ is given in the end of this section.

3.3.1. Solution representation

In a schedule for problem $Rm|r_j,\ LE|^{\#}(C_{max}, U_{cost}, L_{max})$, we are not only concerned about the assignment of jobs (on machines), but also the order of them, since the latter will affect the criterion of L_{max}. Moreover, since we consider learn effect of the machines, different orders of the jobs will also obtain different values of C_{max} and U_{cost}.

Therefore, we adopt an array with $(n + m - 1)$ elements to represent particles' position and velocity, i.e., x_k and v_k $(1 \leq k \leq N)$. In such an array, there are n positive integers from 1 to n to describe jobs, and $m - 1$ negative integers from $-(m-1)$ to -1 to separate the jobs into m sets. The jobs in one set are assigned

to a same machine, and placed following the order in the array. For example, when $m = 3$ and $n = 8$, the array (8, 4, 2, -1, 3, 5, -2, 7, 1, 6) expresses the schedule shown in Figure 1.

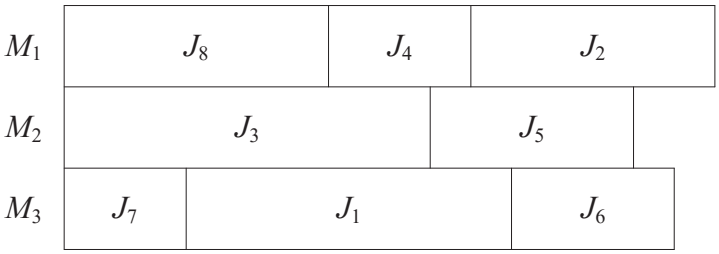

Figure 1. The schedule corresponding to the encoding (8, 4, 2, -1, 3, 5, -2, 7, 1, 6)

Moreover, in Equation 5, R_i ($i = 1$ or 2) is an $(n + m - 1)$ array with random elements of 0 or 1 since it represents the randomicity of the searching process, and the factors w and c_i ($i = 1$ or 2) are all set to be 1 since we consider the previous velocity (v_k^t), the local-best position ($pBest_k^t$) and the global-best position ($gBest^t$) to have the same influence on the velocity of the next generation (v_k^{t+1}).

3.3.2. Operators overloading

In this part, we overload the operators of $-$, $\times$ and $+$ in Equations (5)-(7) so that they can be adaptive for the iterations of our $LFPSO$ algorithm.

(1) Overloading of the operator $-$

Observing from Equation (5), we can find that the subtraction operation is to obtain the difference between the current position and the "best-found" positions (local-best or global-best), so that the next movement can be as close as possible to the "best" ones. Therefore, we define the operation of two arrays $A - B$ as follows. We compare the two arrays by each bit, and if the two elements on the same position have different values, the one at the corresponding position in the result takes the value from array A. Else, we set the corresponding element in the result to be 0. An illustration of two arrays $A - B$ is shown in Figure 2. Note that the result of $A - B$ may be not a feasible solution, which we will adjust in the following operations.

(2) Overloading of the operator $\times$

The subtraction operation remains some partial encoding of the "best-found" solutions, and the purpose of the multiplication operation is to make these encoding randomly changed again. We multiply an $(n + m - 1)$ array R_i ($i = 1$ or 2) with random elements in $\{0, 1\}$ (denoted as A here) and the result of the subtraction operation (denoted as B here) in Equation (5). If the value on one bit in A equals to 1, the element at the corresponding position in the result takes the value of the one

A	8	4	2	-1	3	5	-2	7	1	6
B	8	3	6	4	-1	5	2	7	-2	1
A - B	0	4	2	-1	3	0	-2	0	1	6

Figure 2. An illustration of two arrays $A - B$

in this position in B. Otherwise, the value in A is 0, and the one in the result takes also 0. Figure 3 gives an illustration of two arrays $A \times B$.

A	1	1	0	1	1	0	0	1	0	1
B	0	4	2	-1	3	0	-2	0	1	6
$A \times B$	0	4	0	-1	3	0	0	0	0	6

Figure 3. An illustration of two arrays $A \times B$

(3) Overloading of the operator +

For the addition operation of two arrays, i.e., $A + B$, the result is always a feasible encoding, which tries to keep the characteristics of A and B simultaneously. Observing the Equations (5)-(7), the left operand is always feasible, while the right one could be the results of the previous operations. Therefore, we define the process of $A + B$ as follows: pick out the non-zero elements from array B, and put them on the corresponding positions of the result. Then, we delete these elements from the encoding of array A. Finally, for the "empty" positions in the result, we put the remaining elements in A one by one, as shown in Figure 4.

3.3.3. Initialization and termination

In $LFPSO$, we set the population size to $N = 100$, i.e., there are 100 particles in the system. The initial positions of these particles are generated randomly, and the initial velocities are set to be the same as their positions. Moreover, we claim that a position x_1 dominates x_2 (denoted as $x_1 \preceq x_2$), if the following three inequalities are met simultaneously, and at least one of them is a strict inequality.

$$x_1.C_{max} \leq x_2.C_{max}$$

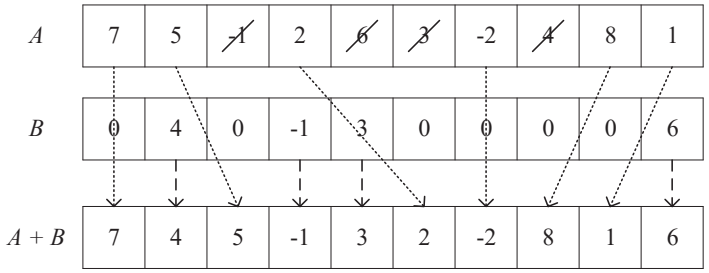

Figure 4. An illustration of two arrays $A + B$

$$x_1.U_{cost} \leq x_2.U_{cost}$$
$$x_1.L_{max} \leq x_2.L_{max}$$

Then, we create a non-dominated set $\mathcal{D}$, and put all the positions that are not dominated by the others (i.e., non-dominated solutions) into $\mathcal{D}$ after initialization.

During the iterations, we update the population according to Equations (5)-(7) with the overloaded operators, and also update set $\mathcal{D}$ if there is a new non-dominated position.

Finally, we terminate our $LFPSO$ if one of the following two conditions are satisfied: (1) The total number of the iterations exceeds 1000 times; (2) The size of $\mathcal{D}$ hasn't been changed for the nearly 50 iterations.

3.3.4. Framework of $LFPSO$

We close this section by proposing the main procedure of our $LFPSO$ algorithm.

$LFPSO$

```
Initialize the population randomly;
Create a non-dominated set D and put all the non-dominated solutions into it;
Set t = 0, pBest_k^0 = x_k^0 for 1 ≤ k ≤ N, and choose arbitrarily one solution
from D as gBest^0;
while (the terminated conditions are not met)
    for each particle P_k
        Update v_k^{t+1} and x_k^{t+1} by Equations (5)-(7) with the overloaded operators;
        Update pBest_k^{t+1} and gBest^{t+1} if they are dominated by the new
population;
        Update set D if there are new non-dominated solutions;
    Set t = t + 1;
Output set D as the Pareto-optimal solution set;
```

4. Computational experiments

In this section, we evaluate our $LFPSO$ through the computational experiments by comparing the one without Levy flight (denoted as PSO here), to indicate the power of this strategy. We first introduce the indicators for evaluating multi-criteria optimization algorithm, and then give the process for generating experiment data, as well as the platform. Finally, the experiment results are reported and analysed in the end of this section.

4.1. Evaluation indicators

We adopt two indicators to compare the qualities of different Pareto-optimal solution sets, which can imply the performances of different algorithms.

(1) C-indicator

C-indicator was proposed by Zitzler et al. [53] to compare two Pareto-optimal solution sets obtained by different algorithms, which is commonly used in the field of multi-criteria optimization. Let $\mathcal{D}_A$ and $\mathcal{D}_B$ be two Pareto-optimal solution sets output by algorithms A and B, respectively, and then we define $C(\mathcal{D}_A, \mathcal{D}_B)$ in Equation 12.

$$C(\mathcal{D}_A, \mathcal{D}_B) = \frac{\left|\{b \in \mathcal{D}_B | \exists a \in \mathcal{D}_A, a \preceq b\}\right|}{|\mathcal{D}_B|} \tag{12}$$

The C-indicator is actually the percentage of solutions in $\mathcal{D}_B$ that are dominated by the ones in $\mathcal{D}_A$. If $C(\mathcal{D}_A, \mathcal{D}_B) = 1$, it means that all the Pareto-optimal solutions generated by algorithm B can be dominated by solutions of algorithm A, so A beats B completely. On the other hand, if $C(\mathcal{D}_A, \mathcal{D}_B) = 0$, any of the solutions in $\mathcal{D}_B$ cannot be dominated by $\mathcal{D}_A$, then we cannot judge that which one is better between the two algorithms.

(2) $NNDS$-indicator

The Number of Non-Dominated Solution ($NNDS$-indicator) is the most direct indicator to measure the quality of a Pareto-optimal solution set. Although we cannot declare that the set with more non-dominated solutions provides better objective values for the considered problems, it offers more choices to the decision-makers for selection.

4.2. Data and platform

We generated the experiment data as follows. Let the number of machines $m \in \{3,\ 5,\ 10\}$, and the number of jobs $n \in \{2m,\ 3m,\ 5m\}$ for small instances, while $n \in \{20m,\ 30m,\ 50m\}$ for big instances. The processing time p_{ij} was taken with

a discrete uniform distribution from $[1,\ 100]$, and the release time r_j was randomly generated from $[0,\ \frac{1}{2m}\frac{\sum_{i=1}^{m}\sum_{j=1}^{n}p_{ij}}{m}]$ [34]. The machine cost u_i followed the discrete uniform distribution of $[5,\ 20]$, and the learning factor $\alpha \in \{-0.152,\ -0.322, -0.515\}$ [12]. All the experiment parameters are listed in Table 1.

Table 1. Experiment parameter setting

Parameter	Symbol	Setting
number of machines	m	$\{3,\ 5,\ 10\}$
number of jobs	n	$\{2m,\ 3m,\ 5m\}$ for small instances $\{20m,\ 30m,\ 50m\}$ for big instances
job processing time	p_{ij}	$[1,\ 100]$
job release time	r_j	$[0,\ \frac{1}{2m}\frac{\sum_{i=1}^{m}\sum_{j=1}^{n}p_{ij}}{m}]$
machine cost	u_i	$[5,\ 20]$
learning factor	α	$\{-0.152,\ -0.322, -0.515\}$

The algorithms were implemented by C++ language in the IDE of Microsoft Visual Studio 2020, and tested on a laptop with Intel Core i7-8550U 1.8GHz CPU and 16GB DDR4 RAM. For each combination of $(m,\ n,\ \alpha)$, 30 random instances were generated, i.e., a total of 1620 tests were executed, and the results are reported and analysed in Section 4.3.

4.3. Results and analyses

We report the experiment results in Tables 2-5, by dividing them into the ones for big or small instances, or related to $(m,\ n)$ or α for statistics. In each table, $\mathcal{D}_{LFPSO}$ and $\mathcal{D}_{PSO}$ denote the final output given by $LFPSO$ and PSO, respectively. Therefore, the C-indicator, $NNDS$-indicator, as well as the running time (in second, s) of the two algorithms, are displayed in the tables.

Table 2. Comparison between $LFPSO$ and PSO on small instances related to $(m,\ n)$

m	n	C-indicator		$NNDS$-indicator		Time (s)	
		$C(\mathcal{D}_{LFPSO}, \mathcal{D}_{PSO})$	$C(\mathcal{D}_{PSO}, \mathcal{D}_{LFPSO})$	$LFPSO$	PSO	$LFPSO$	PSO
3	6	0.8964	0.0939	27.16	13.08	0.0472	0.0623
	9	0.8993	0.0897	39.84	22.57	0.0680	0.0751
	15	0.9069	0.0993	51.03	38.14	0.1094	0.1125
	avg.	0.9009	0.0943	39.34	24.60	0.0749	0.0833
5	10	0.8903	0.0844	50.23	31.07	0.0957	0.0994
	15	0.9002	0.0913	61.49	42.50	0.1308	0.1297
	25	0.9268	0.0958	70.07	51.48	0.1984	0.2102
	avg.	0.9058	0.0905	60.60	41.68	0.1416	0.1464
10	20	0.9229	0.1019	65.59	43.27	0.2350	0.2615
	30	0.9454	0.0963	89.10	61.65	0.3323	0.3553
	50	0.9168	0.0994	117.05	79.66	0.5554	0.5796
	avg.	0.9284	0.0992	90.58	61.53	0.3742	0.3988
AVG.		0.9117	0.0947	63.51	42.60	0.1969	0.2095

Table 3. Comparison between $LFPSO$ and PSO on small instances related to α

α	C-indicator		$NNDS$-indicator		Time (s)	
	$C(\mathcal{D}_{LFPSO}, \mathcal{D}_{PSO})$	$C(\mathcal{D}_{PSO}, \mathcal{D}_{LFPSO})$	$LFPSO$	PSO	$LFPSO$	PSO
-0.152	0.8573	0.1288	63.86	42.65	0.1934	0.2072
-0.322	0.9231	0.0924	62.95	42.53	0.2011	0.2107
-0.515	0.9547	0.0629	63.71	42.63	0.1962	0.2106
AVG.	0.9117	0.0947	63.51	42.60	0.1969	0.2095

Table 4. Comparison between $LFPSO$ and PSO on big instances related to (m, n)

m	n	C-indicator		$NNDS$-indicator		Time (s)	
		$C(\mathcal{D}_{LFPSO}, \mathcal{D}_{PSO})$	$C(\mathcal{D}_{PSO}, \mathcal{D}_{LFPSO})$	$LFPSO$	PSO	$LFPSO$	PSO
3	60	0.9596	0.0696	133.28	78.74	0.8437	1.3208
	90	0.9602	0.0751	141.38	99.47	1.2621	1.9957
	150	0.9617	0.0690	153.21	120.39	2.2005	3.3988
	avg.	0.9605	0.0712	142.62	99.53	1.4354	2.2384
5	100	0.9666	0.0668	155.39	125.23	1.5633	2.3429
	150	0.9707	0.0666	173.58	142.43	2.1637	3.3875
	250	0.9735	0.0588	200.51	152.52	3.6656	5.4354
	avg.	0.9703	0.0641	176.49	140.06	2.4642	3.7219
10	200	0.9678	0.0716	180.21	133.99	3.1679	4.7492
	300	0.9653	0.0615	276.37	223.12	4.9871	7.2303
	500	0.9716	0.0580	353.42	302.72	8.0410	11.9727
	avg.	0.9682	0.0637	270.00	219.94	5.3986	7.9841
AVG.		0.9663	0.0663	196.37	153.18	3.0994	4.6481

Table 5. Comparison between $LFPSO$ and PSO on big instances related to α

α	C-indicator		$NNDS$-indicator		Time (s)	
	$C(\mathcal{D}_{LFPSO}, \mathcal{D}_{PSO})$	$C(\mathcal{D}_{PSO}, \mathcal{D}_{LFPSO})$	$LFPSO$	PSO	$LFPSO$	PSO
-0.152	0.9411	0.1026	195.97	152.65	3.0680	4.6451
-0.322	0.9657	0.0722	197.55	153.38	3.1529	4.6608
-0.515	0.9920	0.0241	195.60	153.51	3.0774	4.6385
AVG.	0.9663	0.0663	196.37	153.18	3.0994	4.6481

Based on these results, we can conclude the differences of $LFPSO$ and PSO, which could imply the power of Levy flight procedure.

(1) From the viewpoint of the solution quality, we claim that $LFPSO$ is much better than PSO, since most of the Pareto-optimal solutions generated by PSO can be dominated by the ones of $LFPSO$ (91.17% for small instances, while 96.63% for big instances). Conversely, the rates that PSO can dominate $LFPSO$ are 9.47% and 6.63%, respectively for small and big instances. (To see the average values of C-indicator in the last rows of Tables 2-5.)

(2) The C-indicator has a strong correlation with the learning factor α, i.e., when α decreases, $C(\mathcal{D}_{LFPSO}, \mathcal{D}_{PSO})$ increases while $C(\mathcal{D}_{PSO}, \mathcal{D}_{LFPSO})$ decreases. This means that the advantages of $LFPSO$ are more obvious if the machines have better learning effect. On the other hand, the relevance between C-indicator and the combination of $(m,\ n)$ (which means different instance scales), is not obvious.

(3) As to the number of non-dominated solution ($NNDS$-indicator), $LFPSO$ also shows its advantage. In each row of the four tables, $LFPSO$ generates more Pareto-optimal solutions than PSO, which indicates that the procedure of Levy flight increases the diversity of the solutions.

(4) There is a clear relationship between the $NNDS$-indicator and the pair of $(m,\ n)$. When the instance scale increases, the number of Pareto-optimal solutions is also increased, indicating that the solution space has more diversity when the problem turns to be big. However, the $NNDS$-indicator has no obvious relation with the learning factor α.

(5) Finally, for the small-scale problem instances, there is no obvious difference in the execution time of the two algorithms. For the large-scale ones, $LFPSO$ runs a bit faster than PSO, indicating that the procedure of Levy flight increases the speed of convergence. Moreover, both of the two algorithms can solve the problem very quickly (i.e., within 10 seconds for the most of the experiments).

5. Conclusion

In this paper, we focused on an unrelated machines scheduling problem with multi-criteria of minimizing makespan, machine cost, as well as the maximal flow-time. Machines have a learning effect so that the actual execution time of a job could be reduced during the processing (compared with its original length). The problem is NP-hard since the classical version on this subject is already NP-hard, which motivated us to propose a particle swarm optimization algorithm with a strategy of Levy flight to solve it. Through the computational experiments, we showed that the procedure of Levy flight could increase the number of Pareto-optimal solutions, and obtain higher-quality solutions than the algorithm without this strategy. Moreover, the introduction of Levy flight can speed up the convergence of the algorithm so that to reduce its running time.

For the future work, we could consider more complex machine environments such as flexible flow shop system, or add more constraints such as setup time, due date, deadline, etc. For the solution approaches, we could investigate other methods to

solve the problem, such as branch-and-bound algorithm, integer programming, etc. Moreover, inspired from the practical applications to create problem models, and then to design algorithms to solve them, will be a very meaningful research topic.

Acknowledgment

This research was partially supported by the Foundation Research Project of the Educational Department of Liaoning Province (No. LJKZZ20220085) and the Cooperation Innovation Plan of Yingkou for Enterprise and Doctor (No. 2022-13).

References

[1] Abedi M., Chiong R., Noman N., and Zhang R. A multi-population, multi-objective memetic algorithm for energy-efficient job-shop scheduling with deteriorating machines. *Expert Systems with Applications*, 157:113348, 2020.

[2] Achugbue J. O. and Chin F. Y. Scheduling the open shop to minimize mean flow time. *SIAM Journal on Computing*, 11(4):709–720, 1982.

[3] Ahmadian M. M., Khatami M., Salehipour A., and Cheng T. C. E. Four decades of research on the open-shop scheduling problem to minimize the makespan. *European Journal of Operational Research*, 295(2):399–426, 2021.

[4] Ali M. Z., Awad N. H., Reynolds R. G., and Suganthan P. N. A balanced fuzzy cultural algorithm with a modified levy flight search for real parameter optimization. *Information Sciences*, 447:12–35, 2018.

[5] Azzouz A., Ennigrou M., and Ben Said L. Scheduling problems under learning effects: classification and cartography. *International Journal of Production Research*, 56(4):1642–1661, 2018.

[6] Bai D., Bai X., Yang J., Zhang X., Ren T., Xie C., and Liu B. Minimization of maximum lateness in a flowshop learning effect scheduling with release dates. *Computers & Industrial Engineering*, 158:107309, 2021.

[7] Bai D., Tang M., Zhang Z.-H., and Santibanez-Gonzalez E. D. Flow shop learning effect scheduling problem with release dates. *Omega*, 78:21–38, 2018.

[8] Bandyopadhyay S. and Bhattacharya R. Solving multi-objective parallel machine scheduling problem by a modified nsga-ii. *Applied Mathematical Modelling*, 37(10-11):6718–6729, 2013.

[9] Biskup D. Single-machine scheduling with learning considerations. *European Journal of Operational Research*, 115(1):173–178, 1999.

[10] Biskup D. A state-of-the-art review on scheduling with learning effects. *European Journal of Operational Research*, 188(2):315–329, 2008.

[11] Caldeira R. H. and Gnanavelbabu A. A pareto based discrete jaya algorithm for multi-objective flexible job shop scheduling problem. *Expert Systems with Applications*, 170:114567, 2021.

[12] Chen X., Chau V., Xie P., Sterna M., and Błażewicz J. Complexity of late work minimization in flow shop systems and a particle swarm optimization algorithm for learning effect. *Computers & Industrial Engineering*, 111:176–182, 2017.

[13] Choobineh F. F., Mohebbi E., and Khoo H. A multi-objective tabu search for a single-machine scheduling problem with sequence-dependent setup times. *European Journal of Operational Research*, 175(1):318–337, 2006.

[14] Dósa G. and He Y. Scheduling with machine cost and rejection. *Journal of Combinatorial Optimization*, 12(4):337–350, 2006.

[15] Garey M. R. and Johnson D. S. *Computers and intractability: A guide to the theory of NP-completeness*. New York, W.H. Freeman & Co., 1979.

[16] Graham R., Lawler E., Lenstra J., and Kan A. R. Optimization and approximation in deterministic sequencing and scheduling: A survey. *Annals of Discrete Mathematics*, 5:287–326, 1979.

[17] Graham R. L. Bounds on multiprocessing timing anomalies. *SIAM Journal on Applied Mathematics*, 17(2):416–429, 1969.

[18] Guo Z., Wong W. K., Li Z., and Ren P. Modeling and pareto optimization of multi-objective order scheduling problems in production planning. *Computers & Industrial Engineering*, 64(4):972–986, 2013.

[19] Haklı H. and Uğuz H. A novel particle swarm optimization algorithm with levy flight. *Applied Soft Computing*, 23:333–345, 2014.

[20] Hematian M., Seyyed Esfahani M. M., Mahdavi I., Mahdavi-Amiri N., and Rezaeian J. A multiobjective integrated multiproject scheduling and multiskilled workforce assignment model considering learning effect under uncertainty. *Computational Intelligence*, 36(1):276–296, 2020.

[21] Hosseinzadeh M., Ghafour M. Y., Hama H. K., Vo B., and Khoshnevis A. Multi-objective task and workflow scheduling approaches in cloud computing: a comprehensive review. *Journal of Grid Computing*, 18:327–356, 2020.

[22] Jensi R. and Jiji G. W. An enhanced particle swarm optimization with levy flight for global optimization. *Applied Soft Computing*, 43:248–261, 2016.

[23] Kennedy J. and Eberhart R. Particle swarm optimization. In *Proceedings of ICNN'95-international conference on neural networks*, volume 4, pages 1942–1948. IEEE, 1995.

[24] Lee W.-C. and Wu C.-C. Minimizing total completion time in a two-machine flowshop with a learning effect. *International Journal of Production Economics*, 88(1):85–93, 2004.

[25] Lei D. Multi-objective production scheduling: a survey. *The International Journal of Advanced Manufacturing Technology*, 43(9):926–938, 2009.

[26] Lei D., Yuan Y., and Cai J. An improved artificial bee colony for multi-objective distributed unrelated parallel machine scheduling. *International Journal of Production Research*, 59(17):5259–5271, 2021.

[27] Li K., Chen J., Fu H., Jia Z., and Wu J. Parallel machine scheduling with position-based deterioration and learning effects in an uncertain manufacturing system. *Computers & Industrial Engineering*, 149:106858, 2020.

[28] Liu D., Tan K. C., Huang S., Goh C. K., and Ho W. K. On solving multiobjective bin packing problems using evolutionary particle swarm optimization. *European Journal of Operational Research*, 190(2):357–382, 2008.

[29] Luo H., Du B., Huang G. Q., Chen H., and Li X. Hybrid flow shop scheduling considering machine electricity consumption cost. *International journal of production economics*, 146(2):423–439, 2013.

[30] Mantegna R. N. Fast, accurate algorithm for numerical simulation of levy stable stochastic processes. *Physical Review E*, 49(5):4677, 1994.

[31] Marinakis Y., Marinaki M., and Migdalas A. A multi-adaptive particle swarm optimization for the vehicle routing problem with time windows. *Information Sciences*, 481:311–329, 2019.

[32] Marler R. T. and Arora J. S. Survey of multi-objective optimization methods for engineering. *Structural and multidisciplinary optimization*, 26(6):369–395, 2004.

[33] Mokhtari H. and Hasani A. An energy-efficient multi-objective optimization for flexible job-shop scheduling problem. *Computers & Chemical Engineering*, 104:339–352, 2017.

[34] Mönch L., Balasubramanian H., Fowler J. W., and Pfund M. E. Heuristic scheduling of jobs on parallel batch machines with incompatible job families and unequal ready times. *Computers & Operations Research*, 32(11):2731–2750, 2005.

[35] Mosheiov G. Parallel machine scheduling with a learning effect. *Journal of the Operational Research Society*, 52(10):1165–1169, 2001.

[36] Mosheiov G. and Sidney J. B. Scheduling with general job-dependent learning curves. *European Journal of Operational Research*, 147(3):665–670, 2003.

[37] Ott A., Bouchaud J.-P., Langevin D., and Urbach W. Anomalous diffusion in "living polymers": A genuine levy flight? *Physical review letters*, 65(17):2201, 1990.

[38] Ozdagoglu G., Erdem S., and Salum L. A special purpose multi-criteria heuristic function for a single machine scheduling problem with forward dynamic programming. *The International Journal of Advanced Manufacturing Technology*, 68(5-8):1875–1886, 2013.

[39] Ozturk O. A truncated column generation algorithm for the parallel batch scheduling problem to minimize total flow time. *European Journal of Operational Research*, 286(2):432–443, 2020.

[40] Poli R., Kennedy J., and Blackwell T. Particle swarm optimization. *Swarm intelligence*, 1(1):33–57, 2007.

[41] Przybylski B. A new model of parallel-machine scheduling with integral-based learning effect. *Computers & Industrial Engineering*, 121:189–194, 2018.

[42] Saber R. G. and Ranjbar M. Minimizing the total tardiness and the total carbon emissions in the permutation flow shop scheduling problem. *Computers & Operations Research*, page 105604, 2021.

[43] Saeedi S., Khorsand R., Bidgoli S. G., and Ramezanpour M. Improved many-objective particle swarm optimization algorithm for scientific workflow scheduling in cloud computing. *Computers & Industrial Engineering*, 147:106649, 2020.

[44] Tian Y., Si L., Zhang X., Cheng R., He C., Tan K. C., and Jin Y. Evolutionary large-scale multi-objective optimization: A survey. *ACM Computing Surveys*, 54(8):174:1–174:34, 2022.

[45] Türkyılmaz A., Şenvar Ö., Ünal İ., and Bulkan S. A research survey: heuristic approaches for solving multi objective flexible job shop problems. *Journal of Intelligent Manufacturing*, 31(8):1949–1983, 2020.

[46] Vahedi Nouri B., Fattahi P., and Ramezanian R. Hybrid firefly-simulated annealing algorithm for the flow shop problem with learning effects and flexible maintenance activities. *International Journal of Production Research*, 51(12):3501–3515, 2013.

[47] Wang X., Gao L., Zhang C., and Shao X. A multi-objective genetic algorithm based on immune and entropy principle for flexible job-shop scheduling problem. *The International Journal of Advanced Manufacturing Technology*, 51(5):757–767, 2010.

[48] Wright T. P. Factors affecting the cost of airplanes. *Journal of the Aeronautical Sciences*, 3(4):122–128, 1936.

[49] Wu M.-C. and Sun S.-H. A project scheduling and staff assignment model considering learning effect. *International Journal of Advanced Manufacturing Technology*, 28(11-12):1190–1195, 2006.

[50] Yagmahan B. and Yenisey M. M. Ant colony optimization for multi-objective flow shop scheduling problem. *Computers & Industrial Engineering*, 54(3):411–420, 2008.

[51] Yeh W.-C., Lai P.-J., Lee W.-C., and Chuang M.-C. Parallel-machine scheduling to minimize makespan with fuzzy processing times and learning effects. *Information Sciences*, 269:142–158, 2014.

[52] Zhang L., Deng Q., Lin R., Gong G., and Han W. A combinatorial evolutionary algorithm for unrelated parallel machine scheduling problem with sequence and machine-dependent setup times, limited worker resources and learning effect. *Expert Systems with Applications*, 175:114843, 2021.

[53] Zitzler E., Thiele L., Laumanns M., Fonseca C. M., and Da Fonseca V. G. Performance assessment of multiobjective optimizers: An analysis and review. *IEEE Transactions on evolutionary computation*, 7(2):117–132, 2003.

Received 21.12.2022, Accepted 16.05.2023