An Improved Particle Swarm Optimization Algorithm for Scheduling Problem on Parallel Machines ⋆

Ning Ding [a], Rongyu Cao [a], Xin Chen [a,*], He Guo [a]

[a] *School of Software Technology, Dalian University of Technology, Dalian 116620, China*

Abstract

In this paper, we study scheduling problem on parallel machines, where each job is unrelated with others and can be processed by any machine, with the goal of minimizing makespan. Once the processing of a job starts, it must be completed without interruption. Instead of classical particle swarm optimization (PSO) algorithm, we propose an improved algorithm with Levy Flight (LPSO). Through the experiments we find that the shortcoming of PSO, i.e., falling into prematurity, has been improved effectively, and the computational results show that the proposed LPSO algorithm is more competitive than the classical one.

Keywords: Scheduling Problem; Parallel Machines; Particle Swarm Optimization; Levy Flight

1 Introduction

The parallel machines scheduling problem can be described as follows: there are m identical machines $\{M_1, M_2, ..., M_m\}$ and n independent jobs $\{J_1, J_2, ..., J_n\}$ which can be processed by any machine. The job is unrelated and cannot be separated or preempted [1]. Furthermore, each machine can only process one job at one time, and there is no precedence relation between jobs. This paper considers the problem of optimal assignment of jobs to machines to minimize the time when the last job has been processed, i.e., the makespan.

Particle swarm optimization (PSO) is a heuristic algorithm that was first proposed by Kennedy and Eberhart in 1995 [2]. It is derived from the observation of feeding habits of birds. When a bird flock is searching for foods, generally, the bird flock is dispersed at first because of the blindness. Every bird will head for the direction of food under the influence of family and society and get the food finally. In this paper, we use the particle swarm optimization (PSO) algorithm, and make the PSO further optimized by joining the method of Levy Flight.

⋆Project supported by the National Nature Science Foundation of China (No.61300016).

*Corresponding author.

Email address: cx.dlut@gmail.com (Xin Chen).

2 Problem description

Each job should be carried out on one of the machines, where the required time for processing job J_i is $p_i (1 \leq i \leq n)$, then the mathematical model can be summarized as follows:

$$\min(\max_{1 \leq j \leq m} (\sum_{i=1}^{n} p_i x_{ij})$$

$$s.t. \sum_{i=1}^{m} x_{ij} = 1$$

$$x_{ij} = \begin{cases} 1 & job\ J_i\ is\ assigned\ to\ machine\ M_j \\ 0 & otherwise \end{cases}$$

Since the problem is NP-hard [3, 4], it is unlikely to obtain the optimal schedule in polynomial time [5]. Therefore, we use heuristic method to solve it in this paper, i.e., an improved PSO algorithm.

3 An improved algorithm named LPSO

3.1 Particle swarm optimization

In Particle swarm optimization (PSO), the particle is the solution vectors of the problem. At first we initialize a set of random solution, then find optimal solution by multi-iterations. In the process of iteration, the particle is affected by two "extreme values" called individual extremum and global extremum. The individual extremum is the optimal solution that every particle has been experienced in multi-iterations, and the global extremum is the optimal solution that all particles have experienced in multi-iterations. Maybe both of the two solutions are not optimal, so we give them two different proportion, and use following rules to describe:

(1) Every particle k is a potential solution, it is initialized by a random value. It begins with a randomized position and randomized velocity. The position and velocity for particle k in the $n - dimensional$ search space are rep-resented by the vectors $x_k = (x_{k1}, x_{k2}, ..., x_{kn})$ and $v_k = (v_{k1}, v_{k2}, ..., v_{kn})$.

(2) Every particle k knows the optimal solution while in the process of iteration, which is the individual extremum recorded as $p_k^t = (p_{k1}^t, p_{k2}^t, ..., p_{kn}^t)$.

(3) Every particle k also knows each other's solution, they can get the optimal solution of other particles in the iterative process, which is the global extremum recorder as $g_k^t = (g_{k1}^t, g_{k2}^t, ..., g_{kn}^t)$.

(4) Every particle k is under the influence of three aspects: current velocity, individual extremum, global extremum, and have the different proportion of relative changes.

This compromise is executed by the following equations at the current iteration of the algorithm:

$$v_k^{t+1} = \omega v_k^t + c_1 r_1 (p_k^t - x_k^t) + c_2 r_2 (g_k^t - x_k^t)$$

$$x_k^{t+1} = x_k^t + v_k^{t+1}$$

Where ω, called the inertia weight, is a constant value chosen by the user to control the impact of the previous velocities on the current velocity. c_1 is the weight given to the attraction to the individual extremum and c_2 is the weight given to the attraction to the global extremum. r_1 and r_2 are uniformly distributed random variables in [0,1]. r_1 and r_2 are knowledge factor, also impact the extremum.

3.2 Levy Flight

The Levy Flight model was proposed by Paul Levy [6]. It's a kind of special random walk which step obeys the Levy distribution [7]. The step l and its appearance probability satisfies the following equation as below [8]:

$$p(l) = cl^{-\mu}, 1 < \mu < 3$$

The research results show that in the case of the food distribute randomly and no information about them, it's the most ideal approach for predator that use Levy Flight to seek food [9].Then, Levy Flight is widely used for the study of animals' behavior. Such as the process of fishing boat, sending and receiving email, searching and rescuing in the earthquake, which are closely linked with Levy Flight.

PSO has a defect that it often runs into prematurity as getting a relatively optimal solution in advance. The solution we get maybe isn't the optimal solution because of premature convergence. But if the velocity of convergence is relatively slow, that will lead to many problems such as too many execution cycles, wasting more times, which will make the algorithm inefficient in getting the optimal solution. In this paper, we use Levy Flight to further optimize the PSO. We not only make the algorithm more efficient through combining the PSO's advantage of rapid convergence and Levy Flight's advantage of random flight to avoid premature convergence, but also increase the searching area to get the optimal approximate solution more accurately and quickly.

In Levy Flight, there're the local search with short step length and search with long step length occasionally. Some solution in search is near the current optimal value, the other solution in search is far away from the current optimal value. So we use Levy Flight in the intelligent optimization algorithm to avoid getting into local optimum. It can expand the scope of the search, increase the diversity of population and jump out of local optimal solution.

3.3 LPSO algorithm

Because the data used in the model is discrete, it's obviously difficult to get the feasible solution by classic equation of continuous PSO algorithm. The process of generating a new position for a selected individual in the swarm is depicted in the following equations [10]:

$$v_k^{t+1} = v_k^t \overset{+}{o}((R_1 \overset{\times}{o}(p_k^t \bar{o})) \overset{+}{o}(R_2 \overset{\times}{o}(g_k^t \bar{o} x_k^t))$$

$$x_k^{t+1} = x_k^t \overset{+}{o} v_k^{t+1}$$

Where R_1 and R_2 are random number between 0 and 1, v_k^t and x_k^t are the kth particle current velocity and position arrays, respectively. p_k^t and g_k^t are the kth particle best position (individual extremum)and the global best position (global extremum) visited so far. The operator of the formula will be described as below:

(1) $\bar{o}$: it means the difference between x_k^t and p_k^t (or g_k^t), x_k^t is a linear array. Each element represents the machine number that the job is working on. It is also used for representing the spatial location of the particle. p_k^t is also a linear array which is the kth particle best position (individual extremum). If $x_{ki}^t \neq p_k^t$, generally speaking, the position of the extreme is more superior to the individual position, so we get the extreme value position. If $x_{ki}^t = p_k^t$, meaning that the current particle in the direction just point to the extreme value position, so there is no need to change the position, the change of the direction vector will be set to 0. We use hamming code to express the different place's number of two arrays when special cases happen. If hamming code is 0, it can be included that $p_k^t = x_k^t$. Then the results should be set as a default, meaning that generating more time sequence making no contribution. This process is shown in Fig.1.

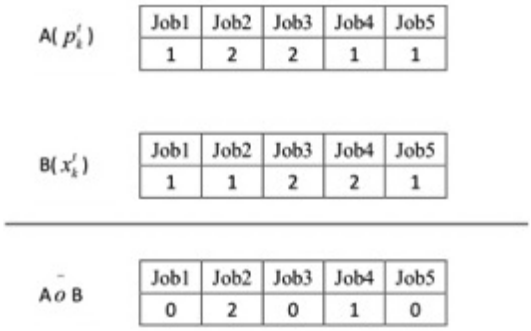

Fig. 1: $\bar{o}$ operator process

(2) $\overset{\times}{o}$: as we can see in the formula, the operand of multiply operation is always a random array of 0, 1 (R_1 or R_2) and the location array ($p_k^t \bar{o} x_k^t$ or $g_k^t \bar{o} x_k^t$), The $\overset{\times}{o}$ operator is equivalent to Hadamard product. The Hadamard product of two $1 \times n$ arrays A and B is denoted by $A \cdot B$ and is an $1 \times n$ array given by $(A \cdot B)_i = a_i \cdot b_i$. Fig. 2 illustrates the manner in which $\bar{o}$ operator performs.

(3) $\overset{+}{o}$: this operator is a crossover operator that typically is used in Genetic algorithms. Two random number rad_1, rad_2 representing the position will be generated, then the crossover will happening between them, i.e., assigning A_{rad_1}, A_{rad_2} to B, and B_{rad_1}, B_{rad_2} assigning to A. Because the two results have same meaning, there is feasible for us to choose one of them randomly as the result of the add operator. This process is shown in Fig.3.

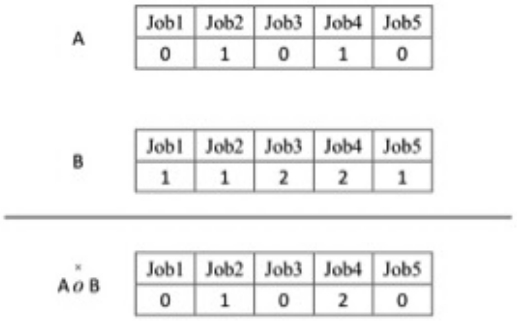

Fig. 2: $\overset{\times}{o}$ operator process

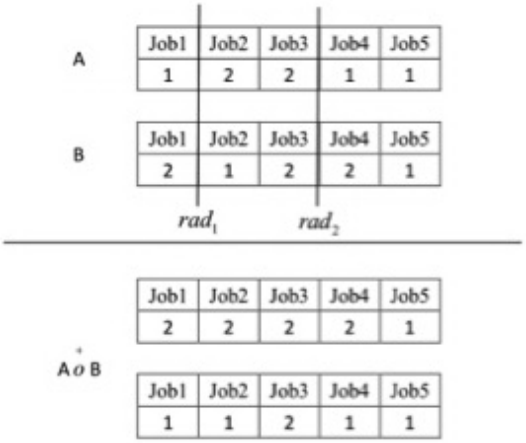

Fig. 3: $\overset{+}{o}$ operator process

The method we proposed is that adding levy flight at the end of each iteration, the main purpose of this method is to avoid falling into local optimal solution of PSO, and random initialization should be executed again, also the main process of the PSO algorithm should be excused. In this paper, the method of levy flight is described as below:

(1) Calculate the Levy Flight's step length l of the current iteration, as the following formula, then go to step 2.

$$l = \frac{MaxGeneration - NowGeneration}{MaxGeneration}$$

Where $MaxGeneration$, is the current iteration number, $MaxGeneration$ is the whole iteration number.

(2) Get the longest numbers of iterations z which the global extremum haven't been updated yet in the current iteration, namely the current numbers of iterations minus the previous numbers whose global extremum have been updated last time. We give the value of maximum tolerances, meaning that at most s generation global extreme value which has not been updated is allowed. Get the current step h using the following formula, then go to step 3.

$$h = \frac{z}{s}$$

(3) Compare l and h. If $l \leq h$, then go to Levy Flight: reinitialize the current position, current velocity vector and the individual extremum, while global extreme value stays unchanged. If $l>h$, then the iteration will be continued.

The general process of LPSO is as follows.

LPSO Algorithm

```
1  initialize particals' velocities and positions
2  while nowgeneration < maxgeneration
3        do calculate l and  k
4           if l ≤ k
5                 do initialize by levy − flight
6           for each particals
7                 do update velocity and positon
8                    update pbest and gbest
9                    if gbest is updated
10                         do reset g = 0
```

4 Results analysis

4.1 Data source

In this section, we investigate a comparison study on the effectiveness of both PSO and LPSO algorithms by solving a large number of problems. We benchmark the experimental frameworks that originally have been designed by Gupta and Ruiz-Tor-res [11], Lee and Massey [12] and Kedia [13]. There are three experimental frameworks, namely E_1, E_2 and E_3 are considered each of them having three influencing variables: the number of machines (m); the number of jobs (n); and the type of discrete uniform distribution used to generate job-processing times (p).

We compare each experimental framework generated by algorithm with the Lower Bound value. The Lower Bound is the minimum optimal solution of parallel machine scheduling problem, which means that there can be no superior solution than the value of the Lower Bound value. The equation of Lower Bound value is $LB = \max(\sum_{i=1}^{n} p_i/n, \max_{i=1}^{n}(p_i))$.

For experimental framework E_1, the number of machines (m) is set at three levels: 3, 4, and 5. When the number of machines is fixed, the number of $jobs(n)$ is set at three levels: $2m$, $3m$, and $5m$. When the number of machines and the number of jobs are all fixed, the processing times are given at two levels, namely, $U(1, 20)$ and $U(20, 50)$. E_1 has been tested total 18 times. For the experimental framework E_2, the number of machines is set at six levels: 2, 3, 4, 6, 8 and 10. When the number of machines is fixed, the number of jobs is set at four levels: 10, 30, 50 and 100. When the number of machines and jobs are all fixed, the job processing times are only generated from $U(100, 800)$. The test intensity of E_2 set is greater than E_1, we mainly observe the results obtained when the job time becomes larger. The experimental framework E_2 has been tested

total 24 times. For experimental framework E_3, the number of machines is set at four levels: 3, 5, 8 and 10. When the number of machines is fixed, the number of jobs is set at six levels: $3m+1$, $3m+2$, $4m+1$, $4m+2$, $5m+1$ and $5m+2$. When the number of machines and the number of jobs are all fixed, the processing times are given at three levels, namely, $U(1,100)$, $U(100,200)$ and $U(200,800)$. E_3 has been tested total 72 times.

We use the control variable method in each experimental framework, under the same times of iterations, each group will test normal particle swarm optimization (PSO) algorithm respectively, as well as Levy Flight particle swarm optimization algorithm (LPSO). We measure our algorithm performance by comparing the ratio between two solutions generated by two algorithms.

For both two algorithms, we use 10 particles to test. We totally got 2000 times iteration cycles. The machine we adopt uses AMD A6 CPU 1.5 GHz and 2GB memory.

4.2 Test results

The results for the experimental framework E_1 are summarized in $table1$, as we can see, there is no significant difference between PSO and LPSO algorithm when the number of machines and particles and processing times are all relatively small. But the LPSO algorithm is still performs slightly better than the common PSO algorithm. The results for the experimental framework E_2 are summarized in $Table$ 2, as the number of machines and the number of particles increases, the advantage of LPSO algorithm appears, and with the processing times increased, the advantage of LPSO become more obvious. The results show the advantage that LPSO algorithm can avoid prematurity and get into individual extremum. The results for the experimental framework E_3 also show that the LPSO algorithm has more advantages. As is summarized in $Table$ 3, LPSO algorithm can obtain a better approximate solution than PSO algorithm, and when the test data become larger, the advantage of LPSO will become more obvious.

4.3 Results analysis

It is observed that, when the number of machines, number of jobs and the random processing times are relatively small, there is not a significant difference between PSO and LPSO algorithm. The reason is that PSO algorithm's performance is good enough in the case of little intensity. Even with the optimization of Levy Flight will not achieve a more optimized condition. But once the PSO algorithm runs into a premature convergence within a certain number of iterations, such like the test of 5 machines and 25 jobs, the processing times are (20, 50), the advantage of LPSO algorithm appears. With the adding of the Levy Flight method, it can jump out of individual extremum easily in the case of premature convergence, and restart searching, get a more optimized result than PSO at last.

When experimental framework E2 with larger intensity is using for the experiment, the advantage of LPSO algorithm appears more obvious, especially in the bigger number of machines and jobs. The results of LPSO algorithm are more optimal than PSO algorithm, which reflects the Levy Flight method can make the disadvantage of PSO algorithm getting into prematurity

Table 1: Results for experiment E1

Experimental Framework E1	m	n	p	PSO	LPSO
–	3	6	U(1, 20)	1.00000	1.00000
–	–	9	–	1.03279	1.03279
–	–	15	–	1.00000	1.00000
–	4	8	U(1,20)	1.17333	1.17333
–	–	12	–	1.02400	1.02400
–	–	20	–	1.01176	1.01176
–	5	10	U(1,20)	1.15385	1.15385
–	–	15	–	1.04651	1.04651
–	–	25	–	1.02273	1.02273
–	3	6	U(20,50)	1.19022	1.19022
–	–	9	–	1.01053	1.01053
–	–	15	–	1.00385	1.00385
–	4	8	U(20,50)	1.09388	1.09388
–	–	12	–	1.03167	1.03167
–	–	20	–	1.01654	1.01053
–	5	10	U(20,50)	1.05505	1.05505
–	–	15	–	1.06498	1.06498
–	–	25	–	1.05085	1.01130
avg.	–	–	–	1.054586	1.052054

Table 2: Results for experiment E2

Experimental Framework E2	m	p	PSO	LPSO
–	2	U(100, 800)	1.00035	1.00035
–	3	–	1.00553	1.00578
–	4	–	1.01386	1.01313
–	6	–	1.06208	1.05281
–	8	–	1.13433	1.12194
–	10	–	1.10843	1.10548
avg.	–	–	1.054097	1.049915

easily was significantly reduced, and make full use of the PSO algorithm's advantage simplicity and efficiency.

But LPSO algorithm also have disadvantage of the limited step length. If the step length is relatively short, which means the frequency of re-initialization will be too high, then the search capability of PSO will be limited. When the PSO algorithm is searching, process will jump out of the search before the algorithm get a more optimized solution, and re-initialization, leading a waste of many times iterative search memory. If the step is too long, LPSO will also getting into prematurity. So the control of step length is essential for LPSO. For different mathematical

Table 3: Results for experiment E3

Experimental Framework E3	m	p	PSO	LPSO
–	3	U(100, 800)	1.00120	1.00189
–	5	–	1.02409	1.02996
–	8	–	1.10089	1.09531
–	10	–	1.16799	1.14359
avg.	–	–	1.073542	1.067688
–	3	U(100,200)	1.01507	1.01526
–	5	–	1.02056	1.01877
–	8	–	1.08835	1.07645
–	10	–	1.13791	1 .12192
avg.	–	–	1.065472	1.058100
–	3	U(1,100)	1.00398	1.00398
–	5	–	1.02898	1.02706
–	8	–	1.10632	1.11622
–	10	–	1.17688	1.15756
avg.	–	–	1.079040	1.076205

model, we need to set different step length to obtain the optimal solution. Which always need a large number of test data.

5 Conclusion

In this paper we use a Levy flight to optimize the particle swarm optimization (PSO) algorithm and to solve the NP-hard problem of parallel machine scheduling problem. Through the experiments we find that PSO has been obvious improved with the optimizing of Levy Flight method. The disadvantage of PSO algorithm falling into prematurity easily has been significantly reduced. Levy Flight can also make other intelligent algorithm to be further optimized, which we will study in the following experiment. And for the Levy Flight, the control of step length is the most important. The superiority of the algorithm is often determined by the step length.

In the future we will use this algorithm to solve different combinatorial optimization problem in the further research, and use the methods of Levy Flight to optimize other intelligent algorithm.

References

[1] H. Tang, and C. Zheng. Scheduling introduction. *Beijing:Science Press* , vol.16, (1990), pp: 1390-1401.

[2] J. Kennedy, and R. Eberhart. Particle Swarm Optimization. *Proceedings of IEEE*, vol.4, (1995), pp: 1942-1948.

[3] Z. Renzhong, and Z. Qianchuan. Discrete event dynamic system. *Tsinghua univercity press*, vol.16, 2001.

[4] P. Brucker. Scheduling Algorithm(The Fifth Edition). *Springer-Verlag.Berlin*, (2007).

[5] M.R. Garey, and D.S. Johnson. Computers and intractability : a guide to the theory of NP-completeness. *San Francisco : Freeman*, (1979).

[6] R. Weron. Levy-stable distributions revisited: tail index > 2 does not exclude the Levy-stable regime. International Journal of Modern Physics C, 2(1995), pp: 209-223.

[7] A.K. Kubala, D. Bana, J. Braziewicz, U. Majewska, and M. Pajek. Concentration distribution of trace elements: from normal distribution to Levy flights. Spectrochimica Acta Part B: Atomic Spectroscopy, vol. 58, (2003), pp. 717-724.

[8] M. Gutowski. Levy flights as an underlying mechanism for global optimization algorithms. arXiv:math-ph/0106003, (2001).

[9] E. Weeks, T. Solomon, J. Urbach, and H. Swinney. Observation of anomalous diffusion and Levy flights. Levy Flights and Related Topics in Physics, (1995), pp: 51-71.

[10] A.H.Kashan and B.Karimi. A discrete particle swarm optimization algorithm for scheduling parallel machines. Computers & Industrial Engineering, 1(2009):216-223.

[11] J. Gupta and A. Ruiz-Torres. A LISTFIT heuristic for minimizing makespan on identical parallel machines. Production Planning & Control, 12(2001):28-36.

[12] C. Lee and J. Massey. Multiprocessor scheduling: Combining LPT and MULTIFIT. Discrete Applied Mathematics, 20:233-242. (1988).

[13] S. Kedia. A job scheduling problem with parallel processors. Technical Report. Ann Arbor, MI: Department of Industrial and Operations Engineering, University of Michigan. (1971)