A Hybrid PSO-CS Algorithm for Parallel Line Job Shop Scheduling To Minimize Makespan

P. Ravichandran, K. Krishnamurthy and R. Parameshwaran

Department of Mechatronics Engineering, Kongu Engineering College, Erode, Tamilnadu, India 638052

Abstract: Parallel line job shop scheduling is the optimal provision and scheduling of jobs in numerous processing lines. Every job is assigned to a specific process line and job is processed to completion in the same specified line. In addition, all jobs allotted to a line are processed in a specific order. The work is focused to find the optimal distribution of jobs to these lines. Also the optimal order of jobs processed in every single line based on distinct processing times and set up times are identified. In this work, parallel line job shop scheduling is done using Cuckoo Search (CS), Particle Swarm Optimisation (PSO) and Hybrid CS-PSO is attempted. A comparison is made in between CS, PSO and Hybrid PSO-CS in solving the scheduling problems. The results show that hybrid PSO-CS algorithm provides a better solution than CS and PSO for parallel line job shop scheduling.

Key words: Particle swarm optimisation (PSO) • Cuckoo Search (CS) • Hybrid PSO-CS • Parallel line job shop scheduling problem

INTRODUCTION

Parallel line job shop scheduling is considered as an advanced version of job shop scheduling problem, it has multiple processing lines. There are a number of machines in each line but the number of jobs may be different for each line. Also the jobs cannot move in between lines. The time, in which the last job is completed in a particular line, is the completion time of that line and it is called as makespan [4]. Thus, the highest makespan among all the lines is considered as the makespan of that process.

As scheduling is very vital in the manufacturing industries, researchers have always shown interest in optimizing the mathematical problem. Al-maamari et al., [1] proposed PSO-CS algorithm for task scheduling problem in cloud computing. The experimental result of cloudbus simulator shows the reduction of the makespan. Bock and hettenhausen [2] proposed PSO algorithm for ontology alignment. It is implemented under the name mapPSO (ontology mapping using PSO). An experimental result shows that proposed PSO is a feasible approach. Chutima and Chimklai [3] proposed a PSO algorithm with negative knowledge (PSONK) to solve the multi-objective two sided mixed model assembly line balancing problems. The result shows that PSONK is a competitive and promising algorithm. Haq et al., [4] discussed the parallel job scheduling using genetic algorithm optimization technique. Randomization algorithm is used to decide the initial set of population. It gave the minimum makespan for constant set up times in each line and also for equal number of machines in each line. Kulkarni and Venayagamoorthy [5] developed a PSO algorithm for Wireless Sensor Networks to reduce the issues such as node deployment, localisation, energy-aware clustering and data aggregation. Moranej and Akhlaghi [6] developed a CS algorithm for optimal distributed generation allocation. It has been implemented to improve the power quality and this algorithm has been compared with PSO and GA. These results indicate that CS is superior to PSO and GA. Rajabioun [7] proposed a new optimization algorithm which is inspired by a cuckoo’s characteristics of egg laying and breeding. CS is found to be superior and gave the minimum makespan. Reddy [8] proposed hybrid of CS and PSO algorithm in order to solve nonslicing floor plan in an efficient manner. The results for benchmark circuits confirmed that the proposed algorithm achieves the optimal result for hard modules placement. Wang et al., [9] proposed an
improved hybrid cooperative algorithm HCCSPSO (Hybrid Cooperative Cuckoo Search Particle Swarm Optimisation) that combines cooperative cuckoo search algorithm and particle swarm optimisation. The result demonstrates the improvement in the efficiency and the effect of the cooperation strategy and the promising of HCCSPSO.

Yang and Deb [10] have proposed the new CS algorithm in combination with Levy flights and employed it for solving structural optimization problems. The CS algorithm is validated first using several benchmark structural engineering problems and found out to be very efficient.

From the earlier studies, this work considers multiple processing lines and CS, PSO and hybrid CS-PSO is used to schedule the jobs in these multiple processing lines.

**Methodology**

**Particle Swarm Optimisation:** Particle swarm optimization (PSO) is a population based heuristic global optimization method and it was developed by Kennedy and Eberhart in 1995. It is developed from swarm intelligence and is based on the research of collective behaviour of bird and fish. When the birds are searching for food, they are either in different direction or at the same direction before they are identifying the place of food is present. While the birds are searching for food from one place to another, one bird among the group can smell the place of food and having the better food resource information. This information is conveyed to all other birds and based on the quality of the information all other birds will update their direction and velocity towards the bird which is having better food resource information. This information is updated velocity is determined with the following equation 2.

\[
V_{i+1} = \omega V_i - c_1 r_1 (P_i - X_i) + c_2 r_2 (G_i - X_i)
\]

\[ (2) \]

where \( \omega \) is the inertia weight that controls the impact of the previous velocity on the current velocity by balancing of exploration and exploitation characteristic of an algorithm. In this work, Linear Decreasing Inertia Weight is considered and it is calculated by the following equation 3.

\[
W_c = W_{max} - \left( \frac{W_{max} - W_{min}}{\text{iter}_{max}} \right) \times \text{iter}
\]

\[ (3) \]

The Linear Decreasing strategy improves the performance of PSO. It is found that Inertia Weight from 0.9 to 0.2 provides the better results. \( c_1 \) and \( c_2 \) represent the weights of the stochastic acceleration effect and if \( c_1 \) is zero there no cognitive element and if \( c_2 \) is zero there no social element in the algorithm. In this paper the value of acceleration parameters \( c_1 \) & \( c_2 \) are taken equal to 2. \( r_1 \) and \( r_2 \) are two random numbers within the range from 0 to 1. In each iteration, the particle move according to the updated velocity and position and the best results of each particle and the whole swarm are updated for the next iteration.

The steps of the PSO technique are given in the following pseudo-code.

**Initialize PSO parameters coefficients, swarmsize**
**Initialize positions, velocities of particles**
**Initialize Personal best position**
**Determine Best particle of the swarm**
**Loop**
  **Update velocities**
  **Update positions**
  **Determine personal best position**
  **Determine Best particle of the swarm**
  **Local search (optional)**
**Until stop criterion**

The initial positions and velocities are determined randomly. The coefficients are chosen according to the earlier mathematical study. The comparison between two particles is done with the fitness value. The stop criterion is the maximum number of iteration or the computational time.
Cuckoo Search Algorithm: Cuckoo search algorithm is a newly established evolutionary algorithm based on the breeding manners of the cuckoo bird. Like GA and PSO, it is also a population centered algorithm developed by Yang and Deb in 2009. The main concept of cuckoo search algorithm explained by Yang and Deb [9] are, Cuckoo lays one egg at a time and dumps it in a randomly chosen nest, the best nests with high quality eggs will be retained to the next generation, the number of host nests are fixed and a host bird can discover an alien egg with a lesser probability. In this case, the host bird either throws the egg away or abandons the nest so as to build a completely new nest in new location.

Here, the nests are the initial population and the egg is the solution. The best solution (egg) is obtained by performing limited number of iterations. An objective function or the fitness function is the matured cuckoo in this concept. This concept was mainly developed for minimization problems. Hence, minimum the fitness value better is the optimization. Thus this concept is also used for maximization problems. For a maximization problem, the solution is proportional to the fitness value. This algorithm is even more enhanced by Levy flight random walk. Levy flight is a kind of random walk which is based on the flight characteristic behavior of the animal. The basic concept of cuckoo search can be plotted as the pseudo code as shown below.

Start
Objective function \( f(x) \), \( x = (x_1, \ldots, x_d) \)
Generate initial population of \( n \) host nests \( x_i \) (i = 1, 2, \ldots, n)
While \( t < \text{MaxGeneration} \) or (stop criterion)
Get a cuckoo randomly by Levy flights evaluate its quality/fitness \( F_i \)
Choose a nest among \( n \) (say, \( j \)) randomly
If \( (F_i > F_j) \),
replace \( j \) by the new solution;
end
A fraction \((p)\) of worse nests are abandoned and new ones are built;
Keep the best solutions (or nests with quality solutions);
Rank the solutions and find the current best
end while
Postprocess results and visualization
end

Next, the processing times and the makespan of all the lines are calculated. The required objective function is formulated and imposed in to the program. 25 initial nests (population) are generated randomly (as the initial population (\( n \)) is taken as 25), each nest having a dimension of 10. Each nest has different job sequence in it. The objective function, the makespan, of each nest is calculated using the processing times and the setup times given in Table 2 and 3 respectively. A random job sequence (random cuckoo) is generated and its objective function is also calculated. The makespan value of the calculated objective function of the random cuckoo is then compared with the makespan value of randomly chosen nest.

If the makespan value of the cuckoo is less when compared to that of randomly chosen nest, then, the cuckoo replaces the value of the randomly generated nest, it else remains the same and continues to go for the next iteration. The worst nests are removed (due to the parameter called discovery rate of alien egg (\( p_a \)) which is given as 0.25) and is replaced by random generation of nests. Solutions are ranked accordingly and the cuckoo is updated via Levy flights and thus the same procedure is repeated again and again till the iteration number is reached. Thus in cuckoo search, unlike GA and PSO, the number of input parameter is limited only to 2. They are: \( n \), number of nests and \( p_a \) discovery rate of alien eggs. Here an initial population of 25 nests and the discovery rate of alien eggs as 0.25 have been taken.

In the objective function, the makespan of the three lines is compared in order to get the maximum makespan and that makespan is considered to as the optimal makespan of this problem. The job sequence (which is rather known as nest) in finding the optimal makespan is the optimal job sequence of the problem.

Hybrid PSO-CS Algorithm: Hybrid Algorithm is formed by combining both cuckoo search and particle swarm optimization. It is combined in a way where, cuckoo search optimizes and the result of CS algorithm is taken as the initial velocity of PSO algorithm and the process of optimization continues till the maximum limit is reached. The flow diagram of Hybrid PSO-CS is shown in Fig. 1.

Problem Description: The parallel line job shop scheduling problem in this work is taken from Haq et al [4]. It consists of three lines, each with three machines. The assumptions made here are as follows:
All processing lines contain equal number of machines

The setup time of a job is same for all lines with respect to another job

Any job can be processed in any line

Machine breakdown is not permitted

Machines in each line are same in the identity as well as in the order.

Thus it finds the optimal job scheduling with minimum makespan. The benchmark problem [4] has 10 numbers of jobs, 3 identical machines on each lines, the total number of lines are 3. The setup times, processing times and the quantity of jobs are given in Table 1, 2 and 3 respectively.

The processing time of each job in each line is calculated. For example, if the order of jobs in the processing line 1 is J2, J3 and J4, then the processing time is calculated using equation 4.
Table 4: Result of CS algorithm for the sample problem

<table>
<thead>
<tr>
<th>Iteration number</th>
<th>Lines</th>
<th>Sequence</th>
<th>Makespan</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>L1</td>
<td>4–5–7–1</td>
<td>3706</td>
</tr>
<tr>
<td></td>
<td>L2</td>
<td>10–2–9</td>
<td></td>
</tr>
<tr>
<td></td>
<td>L3</td>
<td>8–3–6</td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>L1</td>
<td>5–2–9–1</td>
<td>3573</td>
</tr>
<tr>
<td></td>
<td>L2</td>
<td>7–6–3</td>
<td></td>
</tr>
<tr>
<td></td>
<td>L3</td>
<td>4–8–10</td>
<td></td>
</tr>
<tr>
<td>50</td>
<td>L1</td>
<td>2–1–5–6</td>
<td>3450</td>
</tr>
<tr>
<td></td>
<td>L2</td>
<td>4–3–7</td>
<td></td>
</tr>
<tr>
<td></td>
<td>L3</td>
<td>8–9–10</td>
<td></td>
</tr>
<tr>
<td>75</td>
<td>L1</td>
<td>1–5–2–6</td>
<td>3443</td>
</tr>
<tr>
<td></td>
<td>L2</td>
<td>4–7–3</td>
<td></td>
</tr>
<tr>
<td></td>
<td>L3</td>
<td>8–9–10</td>
<td></td>
</tr>
<tr>
<td>100</td>
<td>L1</td>
<td>1–6–2–5</td>
<td>3441</td>
</tr>
<tr>
<td></td>
<td>L2</td>
<td>4–3–7</td>
<td></td>
</tr>
<tr>
<td></td>
<td>L3</td>
<td>10–8–9</td>
<td></td>
</tr>
</tbody>
</table>

Processing Time = (Processing time of job 2 in line 1) + (Processing time of job 3 in line 1) + (Processing time of job 4 in line 1)  \( (4) \)

Thus, the calculated processing time is added with their respective setup times from Table 4. Consider the order of the jobs to be J2 followed by J3 and J4. Now the makespan for the above example can be calculated as shown below equation 5.

\[
\text{Makespan} = \text{Processing time} + \text{Setup time (1,2)} + \text{Setup time (3,4)} \quad (5)
\]

Thus, the makespan of all lines are calculated. The highest makespan of the three lines is considered as the makespan of the entire process.

RESULTS AND DISCUSSION

The optimisation procedures developed in this work based on the two approaches CS and PSO and combination of these two approaches. These approaches have been implemented using MATLAB and is made to run for different iterations. The program is made to run for 100 iterations and the results are compared. The results of CS, PSO, hybrid PSO-CS and its respective iteration number are shown in Table 4, 5 and 6 respectively.

The comparison of results between CS, PSO and hybrid PSO-CS is shown in Table 7. The results shows that hybrid PSO-CS algorithm has the minimum makespan value and is found out to be 3437 units. The same problem solved using CS produces minimum makespan of 3441 time units and PSO produces only a minimum makespan of 3459 time units and thus when compared, hybrid PSO-CS outperformed CS and PSO, produced 4 time units lesser than that of CS and 27 time units lesser than that of PSO for the same number of iterations. Thus, Hybrid PSO-CS is superior to CS and PSO.

CONCLUSION

Swarm intelligence-based algorithms such as cuckoo search and particle swarm optimization are very efficient in solving a wide range of nonlinear optimization problems. These two algorithms and hybrid of PSO-CS are implemented successfully for solving the scheduling
optimization problem. Results obtained from these approaches are compared and the performances are analysed. Hybrid PSO-CS is found to be superior and gives minimum makespan.

REFERENCES


