PARTICLE SWARM OPTIMIZATION BASED ON BOTTLENECK MACHINE FOR JOBSHOP SCHEDULING

Rahmi Maulidya

Industrial Engineering Department, Industrial Engineering Faculty, Trisakti University
rmauli@yahoo.com

ABSTRACT

In job shop production system, bottleneck machine is treated as a center of production planning so it is important to maximize the production capacity. Bottleneck scheduling problem divided solution into forward and backward scheduling. The completion time of the job until bottleneck machine is a due date for all job and the completion time of the job after bottleneck machine is the real completion time for all job. In most scheduling problem, solution using Particle Swarm Optimization is only to minimize makespan in forward scheduling or minimize the number of tardy job in backward scheduling. This research present the methodology to solve job shop scheduling problem based on bottleneck machine using Particle Swarm Optimization.

Keywords : scheduling, job shop, bottleneck, Particle Swarm Optimization, due date, makespan

1. INTRODUCTION

Job shop scheduling is characterized by the routing of each products. n job are processed to completion on m unrelated machines. The routing and processing time is known in advance. Processing time are fixed. Machine is available from time zero and operations are processed without preemption. Job shop scheduling is strongly NP-hard and solve using metaheuristic algorithm. Genetic algorithm, simulated annealing algorithm, ant colony algorithm and particle swarm optimization is classified into metaheuristic.

Particle swarm optimization (PSO) is one of the latest evolutionary optimization methods inspired by nature that includes evolutionary strategy (ES), evolutionary programming (EP), genetic algorithm (GA), and genetic programming (GP). PSO is based on the metaphor of social interaction and communication such as bird flocking and fish schooling. PSO is distinctly different from other evolutionary-type methods in a way that it does not use the filtering operation (such as crossover and/or mutation), and the members of the entire population are maintained through the search procedure so that information is socially shared among individuals to direct the search towards the best position in the search space (Tasgetiren, 2007).

Job shop scheduling problem can also be solved by using bottleneck scheduling. Bottleneck machine is defined using the longest processing time of machine. In bottleneck scheduling, there is forward scheduling and backward scheduling, backward scheduling is used until bottleneck machine and forward scheduling is used after bottleneck machine to minimize makespan. PSO mostly use to minimize makespan in forward scheduling. Although there is PSO use to minimize due date, but there is no PSO use in both side.

This paper present the methodology to solve job shop scheduling problem using Particle swarm optimization based on bottleneck machine.

2. LITERATURE

2.1. Particle Swarm Optimization

Particle Swarm Optimization (PSO) in introduced as a method for nonlinear optimization. This method uses a model of flight of birds (or motion of particles) to solve the optimization problem in which any potential solution in search space is considered as a potential position for particles. Swarm of particles move through search space under a defined dynamics of flight and find the best solution as the optimum solution. The position of i-th particle
is presented by an n-dimensional vector for an n-dimensional search space (Rezaie, 2010).

Algorithm of PSO method has steps below (Rezaie, 2010):
1) Problem definition: This step includes the definition of cost function and determining the range of search space. The minimum and maximum values which the particles could take as coordinates of their position should be determined.
2) Initialization: Initialize positions and velocities of particles randomly. Also, best position of each particle is set to be its initial position and global best position is set to be the best initial position associated with cost function.
5) Repeat: Go to step 3 again and repeat till a stopping criterion is satisfied. The stopping criterion could be considered as taking a predefined number of iterations.

2.2. Notation

\( Z_i \) : random number n in particle i and job \( j \)

\( U_{ij} \) : uniform number in particle i and job \( j \)

\( X_{ij} \) : particle i in swarm at iteration \( t \)

\( x_{i,j}^t \) : value position for particle \( i \) and dimension \( j \) where \( (j = 1,2,\ldots,n) \) and \( n \) is the number of dimension. So, initial iteration is noted as \( X_i^0 = [x_{i,1}^0, x_{i,2}^0, \ldots, x_{i,n}^0] \)

\( v_{ij}^t \) : velocity for particle \( i \) and job \( j \) at iteration \( t \)

\( \pi_{ij}^t \) : the number of job \( j \) for particle \( i \) in permutation at iteration \( t \)

\( NP \) : the number of particle where the amount is twice of number of dimension (job)

\( f_i^t \) : makespan of fitness value in particle \( i \) and iteration \( t \)

\( f_{pb} \) : function of the best makespan in personal best

\( f_{gb} \) : function of the best makespan in global best

\( P_i^t \) : Personal Best at iteration \( t \) and particle \( i \)

\( p_{ij}^t \) : Personal Best at particle \( i \), job \( j \) and iteration \( t \)

\( G_i^t \) : Global Best for iteration \( t \)

\( g_{ij}^t \) : Global Best for particle \( i \) and job \( j \) at iteration \( t \)

\( w^t \) : Inertia Weight at iteration \( t \)

\( c_1, c_2 \) : coefficients acceleration where \( c_1, c_2 = 2 \)

\( r_1, r_2 \) : uniform random number from 0 until 1 where \( r_1 \) is uniform number for particle position and \( r_2 \) is uniform number for particle velocity. \( r_1 \) and \( r_2 \) are uniform number from \( U_{ij} \)

\( T_i^t \) : Tardy for particle \( i \) and iteration \( t \)

The steps for PSO are:

**Step 1 Initialization**

a. Generate random number \( r_1 \) using LCG

\[ Z_i = (aZ_{i-1} + b) \pmod{m} \quad (1) \]

\[ U_i = Z_i / m \quad (2) \]

b. Set \( t = 0 \), and \( NP = \) twice number of dimension

c. Define random particle as \( \{ X_i^0, i = 1,2,\ldots,NP \} \) where

\[ X_i^0 = [x_{i,1}^0, x_{i,2}^0, \ldots, x_{i,n}^0] \]

using equation

\[ x_{ij}^0 = x_{\text{min}} + (x_{\text{max}} - x_{\text{min}}) \ast r_i \]

where \( x_{\text{min}} = 0.0 \), \( x_{\text{max}} = 4.0 \) and \( r_i \) is uniform random number between 0 and 1

d. Define initial velocity for each particle i randomly

\[ v_{ij}^0 = v_{\text{min}} + (v_{\text{max}} - v_{\text{min}}) \ast r_2 \]

where \( v_{\text{min}} = -4 \) and \( v_{\text{max}} = 4 \), and \( r_2 \) is define from uniform random number between 1 and 0.

e. Generate permutation using Smallest Position Value (SPV) for particle \( X_i^0 \)

\[ \pi_i^0 = [\pi_{i,1}^0, \pi_{i,2}^0, \ldots, \pi_{i,n}^0] \]

(5)

d. Evaluate every particle i in swarm to look for makespan of every particle noted as function \( f_i^0 \) where \( i = 1,2,\ldots,NP \)

e. For every particle in swarm, set

\[ P_i^0 = X_i^0, \quad p_{ij}^0 = [p_{i,1}^0, p_{i,2}^0, \ldots, p_{i,n}^0] \]

(6)
g. Define the best fitness value which means the best makespan from every particle in swarm. Makespan is noted with:

\[ f_i = \min \{ f_{i,0}, i = 1,2, \ldots, NP \} \tag{7} \]

h. Set global best or Gbest

\[ G_{i,0}^0 = x_{i,0}^0, G_{i,1}^0 = [g_{i,1} = x_{i,1}, \ldots, g_{i,n} = x_{i,n}] \]

i. is particle. \( G_{i,0}^0 \) is value of Global Best at initial iteration untill Global Best equal to the value of particle position.

**Step 2 Update the iteration**

\[ t = t+1 \tag{9} \]

**Step 3 Update Inertia weight**

Update inertia weight with

\[ w^t = w^{t-1} \times \beta \tag{10} \]

where \( \beta \) is decrement factor with value 0.975 and \( w^0 = 0.9 \)

**Step 4 Update Velocity**

\[ v^t_i = w^{t-1} v^{t-1}_i + c_1 r_1 (p^t_i - x^{t-1}_i) + c_2 r_2 (g^t_i - x^{t-1}_i) \tag{11} \]

Where c is coeffisien corelation. For \( p^t_i \) and \( g^t_i \) suite to previous iteration.

**Step 5 Update Position**

\[ x^{t+1}_i = x^{t-1}_i + v^{t}_i \tag{12} \]

**Step 6 Define Permutation**

Using SPV to define new permutation:

\[ \pi^t_{i,1} = [\pi^t_{i,1}, \pi^t_{i,2}, \ldots, \pi^t_{i,m}] \tag{13} \]

**Step 7 Update Personal Best**

Each particle is evaluated using permutation. Permutation can be renew if

\[ f^t_i < f^{pb}_i, i = 1,2, \ldots, p \tag{14} \]

so personal best can be changed into \( P^t_i = X^t_i \) and \( f^{pb}_i = f^t_i \), which means if makespan in the next iteration is smaller atu equal to makespan at iteration \( t-1 \), \( P^t_i \) is changed into the value of particle at the best iteration, so that for personal best.

**Step 8 Update Global Best**

Find minimum value of Global Best

\[ f^t_i = \min \{ f^{pb}_i \} \tag{15} \]

i = 1,2,\ldots, NP

it means the best makespan of all particle, so global best is renewed into \( G^t_i = X^t_i \) and

\[ f^{gb} = f^t_i \]

**Step 9 Stopping Criterion**

If iteration exceed maximum number of iteration, or exceed maximum time, the iteration will be stop.

### 3. METHODOLOGY

Particle Swarm Optimization start with generating random number to define the number of particle, the petition of particle, and particle velocity. The number of particle is set two times from the number of job. Permutation ranking is set using SPV. Job is schedule using permutation ranking.

In this research, job is scheduled based on the position of bottleneck machine. For position before bottleneck machine, define particle due date and generate sequence using backward scheduling. For position after bottleneck machine, sequence is defined using forward scheduling. Both result can define personal best position with set of \( P^t = X^t_i \) for each particle. Next step is the same for PSO algorithm which define Global Best \( G^t = X^t_i \).

The sequence is set based on global best and renew at every iteration until smallest or constant global best. In every renew iteration, it is followed by the renew of weight.

### 4. NUMERICAL EXAMPLE

In this paper, the data is set in table 1. The processing time and the routing is fixed and is known in advance.

<table>
<thead>
<tr>
<th>JOB</th>
<th>Processing time</th>
<th>Routing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operation</td>
<td>Operation</td>
<td></td>
</tr>
<tr>
<td>J1</td>
<td>5</td>
<td>10</td>
</tr>
<tr>
<td>J2</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>J3</td>
<td>9</td>
<td>6</td>
</tr>
<tr>
<td>J4</td>
<td>7</td>
<td>5</td>
</tr>
</tbody>
</table>

The total processing time of each machines show that machine B is the longest. Machine B is also a bottleneck machine. Machine B will be scheduled for the first time.

<table>
<thead>
<tr>
<th>JOB</th>
<th>machine</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
<td>C</td>
</tr>
<tr>
<td>1</td>
<td>5</td>
<td>10</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>8</td>
</tr>
<tr>
<td>3</td>
<td>7</td>
<td>6</td>
</tr>
<tr>
<td>4</td>
<td>11</td>
<td>7</td>
</tr>
<tr>
<td>Total</td>
<td>27</td>
<td>31</td>
</tr>
</tbody>
</table>
Now, focus is on machine B as bottleneck machine,

![Routing Product](image)

Figure 1. routing product

Random number is generate using this data:
- a : multiplier = 23
- b : increment factor = 0
- m : modulus = 100000001
- \( Z_0 : 47594118 \)

Random number will result \( r_1 \) dan \( r_2 \), then it is used to define permutation in table 3.

<table>
<thead>
<tr>
<th>Particle</th>
<th>Sequence</th>
</tr>
</thead>
<tbody>
<tr>
<td>X_0^1</td>
<td>J2 J3 J4 J1</td>
</tr>
<tr>
<td>X_0^2</td>
<td>J1 J3 J2 J4</td>
</tr>
<tr>
<td>X_0^3</td>
<td>J4 J1 J2 J3</td>
</tr>
<tr>
<td>X_0^4</td>
<td>J2 J3 J4 J1</td>
</tr>
<tr>
<td>X_0^5</td>
<td>J1 J4 J2 J3</td>
</tr>
<tr>
<td>X_0^6</td>
<td>J1 J3 J4 J2</td>
</tr>
<tr>
<td>X_0^7</td>
<td>J3 J1 J2 J4</td>
</tr>
</tbody>
</table>

Each permutation in \( t=0 \) is observed to get makespan for initial. Due date is set until machine bottleneck. Due date is the longest time until machine bottleneck. Particle which give the best result is J4-J1-J3-J2. Gantt chart for the sequence is in figure 2.

![Gantt Chart](image)

Figure 2. Gantt chart

Table 3. Sequence Partikel \( X_0^1 \) – \( X_0^7 \)

<table>
<thead>
<tr>
<th>Particle</th>
<th>Sequence</th>
</tr>
</thead>
<tbody>
<tr>
<td>X_0^1</td>
<td>J2 J3 J4 J1</td>
</tr>
<tr>
<td>X_0^2</td>
<td>J1 J3 J2 J4</td>
</tr>
<tr>
<td>X_0^3</td>
<td>J4 J1 J2 J3</td>
</tr>
<tr>
<td>X_0^4</td>
<td>J2 J3 J4 J1</td>
</tr>
<tr>
<td>X_0^5</td>
<td>J1 J4 J2 J3</td>
</tr>
<tr>
<td>X_0^6</td>
<td>J1 J3 J4 J2</td>
</tr>
<tr>
<td>X_0^7</td>
<td>J3 J1 J2 J4</td>
</tr>
</tbody>
</table>

Table 4. Result

<table>
<thead>
<tr>
<th>PSO proposed</th>
<th>Makespan</th>
<th>Flowtime</th>
<th>Idle Time</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>32</td>
<td>19,5</td>
<td>A = 5</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>B = 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>C = 3</td>
</tr>
</tbody>
</table>

5. CONCLUSION

The PSO based on bottleneck machine is effective for large number of job. Testing the PSO into flowhop scheduling is the next research and define the bottleneck machine using theory of constraint.

6. REFERENCES

(d) Divani, Vania. (2012). Usulan Penjadwalan Job Dengan Metode Particle Swarm Optimization Untuk Meminimasi Number Of Tardy Jobs Di PT, Faco Global Engineering. Universitas Trisakti, Jakarta
(i) Tasgetiren, M. F., Liang, Y.-C., Sevkli, M., dan Gencyilmaz, G. (2007). “A Particle Swarm Optimization Algorithm for Makespan and Total Flowtime Minimization in Permutation Flowshop Sequencing Problem”. European Journal...
AUTHOR BIOGRAPHIES

Rahmi Maulidya is a lecturer in Department of Industrial Engineering, Faculty of Industrial Technology, Trisakti University, Jakarta. She received her Master of Industrial Engineering from Institut Technology Bandung in 2005. Her research interests are in the area of Production Planning & Control, Robotic and Design Product. She is a member of the Production System Laboratory. Her email address is <rmauli@yahoo.com>