Particle Swarm Optimization Approach for Scheduling of Flexible Job Shops

[1]Srinivas P. S., [2]Ramachandra Raju V., [3]C.S.P Rao.
[1]*Associate Professor, V. R. Siddhartha Engineering College, Vijayawada*

[2]*Professor, Department of Mechanical Engineering,*
College of Engineering, JNTUK, Vizianagaram,
[3]*Professor, Department of Mechanical Engg, NIT, Warangal*

Abstract

Recent developments in research on decision making have linked up the strings of optimization with the social behavior of the insects. There have been a host of complex problems like scheduling, project management, routing etc., that are conveniently mapped with natural environment and solved through the inspiration of the insects. This paper illustrates the application of particle swarm optimization (PSO) approach for solving a simple and flexible job shop problems with an objective of minimizing the maximum completion time of all the jobs. The preliminary results are quite encouraging and motivating for the researchers to use PSO as a powerful tool in real time scheduling problems.

Keywords: *Flexible Job Shop, Particle Swarm Optimization, NP-hard, Swarm Intelligence*

1. Introduction

Scheduling is concerned with allocating limited resources to tasks to optimize some performance criterion, such as completion time or production cost. Scheduling of a job shop is very important in both fields of production management and combinatorial optimization. However, it is quite difficult to achieve an optimal solution to this problem with traditional optimization approaches owing to the high computational complexity. A large number of approaches to the modeling and solution of these scheduling problems have been reported in the Operations Research (OR) literature, with varying degrees of success.

The flexible job shop scheduling problem (FJSP) is one of the hardest combinatorial problems and very difficult to solve. FJSP is an extension of the classical JSP which allows an operation to be processed by any machine from a given set. It incorporates all the difficulties and complexities of its predecessor JSP and is more complex than JSP because of the addition need to determine the assignment of operations to machines. In most of its practical formulations, the FJSP is known to be NP-hard, so exact solution methods are unfeasible for most problem instances and heuristic approaches must therefore be employed to find good solutions with reasonable search time. In this paper, PSO based solution methodology is adopted for solving FJSP.

2. Literature

Kennedy J, Eberhart R C [3] are the first persons introducing this particle swarm optimization. It is an evolutionary computation technique mimicking the behavior of flying birds and their means of information exchange. It combines local search (by self experience) and global search (by neighboring experience), possessing high search efficiency. Chandrasekaran. S et al [2] dealt the problem of scheduling in flow shops with the objective of minimizing makespan, total flow time and completion time variation. Rahimi [4] considered a bi-criteria permutation flow shop scheduling problem, where weighted mean completion time and weighted mean tardiness are to be minimized simultaneously. Since a flow shop scheduling problem has been proved to be NP-hard in strong sense, an effective multi-objective particle swarm (MOPS), exploiting a new concept of the Ideal Point and a new approach to specify the superior particle's position vector in the swarm, is designed and used for finding locally Pareto-optimal frontier of the problem.

Zhixiong Liu [7] proposed the particle representation based on operation-particle position sequence. In the particle representation, the mapping between the particle and the scheduling solution is

established through connecting the operation sequence of all the jobs with the particle position sequence. The particle representation can ensure that the scheduling solutions decoded are feasible and can follow the particle swarm optimization algorithm model. Weijun Xia and Zhiming Wu [6] proposed a hybrid optimization approach for multi objective flexible job shop scheduling problems, where they can integrate particle swarm with simulated annealing for solving job shop problems. By reasonably hybridizing these two methodologies, they develop an easily implemented hybrid approach for the multi-objective flexible job-shop scheduling problem (FJSP).

Sha and Cheng [5] proposed a hybrid particle swarm optimization (PSO) for the job shop problem. Since the solution space of the JSP is discrete, we modified the particle position representation, particle movement, and particle velocity to better suit PSO for the JSP. They modified the particle position based on preference list-based representation, particle movement based on swap operator, and particle velocity based on the tabu list concept in our algorithm. Giffler and Thompson's heuristic is used to decode a particle position into a schedule. Furthermore, they applied tabu search to improve the solution quality.

Ajith Abraham et al [1] introduced a hybrid metaheuristic, the Variable Neighborhood Particle Swarm Optimization (VNPSO) algorithm, consisting of a combination of the Variable Neighborhood Search (VNS) and Particle Swarm Optimization (PSO). The proposed VNPSO algorithm is used for solving the multi-objective Flexible Job-shop Scheduling Problems (FJSP).

3. Particle Swarm Optimization

Particle swarm optimization (PSO) is a popular problem solving technique in the swarm intelligence (SI) paradigm. It was first introduced by Kennedy and Eberhart in 1995. They developed simple methods which could efficiently optimize continuous nonlinear mathematical functions. Borrowing ideas from artificial life (A-life), social psychology and swarming theory, PSO simulates swarms such as flocks of birds and schools of fish searching for food.

Also, PSO is related to evolutionary computation (EC), but it is somewhat different. Similar to many EC techniques, PSO initializes a problem state to a population of randomly distributed solutions. Unlike many other ECs however, PSO "evolves" solutions based on individual experience and group experience, rather than using evolutionary operators (e.g. the crossover and the mutation operators in genetic algorithms). It assumes that socially shared information helps its population evolve. In other words, the population iteratively updates and searches for optima with the shared information. In this paper, PSO is employed to solve the flexible job shop scheduling problem with an objective of minimal make span.

In this paper we use the global model equations as follows (Shi & Elbert, 1999):

$$V_{id} = W * V_{id} + C1 * Rand * p_{id} - X_{id} + C2 * rand * p_{gd} - X_{id} \quad (1)$$

and $$X_{id} = X_{id} + V_{id} \quad (2)$$

where V_{id} is called the velocity for particle I, represents the distance to be traveled by this particle from its current position, X_{id} represents the particle position, p_{id} which is also called as *pbest* (local best solution), represents i[th] particles best previous position, and p_{gd}, which is also called *gbest* (global best solution), represents the best position among all particles in the swarm. W is the inertial weight. It regulates the trade-off between the global exploration and local exploitation abilities of the swarm. C1 and C2 represent the weight of the stochastic acceleration terms that pull each particle toward *pbest* and *gbest* positions. $Rand$ and $rand$ are two random functions in the range [0,1].

The inertia weight is set using the following equation

$$W = W_{\max} - \frac{W_{\max} - W_{\min}}{iter_{\max}} iter,$$

Where

$W_{\max}$ = initial value of the weighting coefficient

$W_{\min}$ = final value of the weighting coefficient

$iter_{\max}$ = maximum number of iterations

$iter$ = current iteration or generation number

The process of implementing the PSO algorithm is as follows.

1. Initialize a swarm of particles with random positions and velocities in the D-dimensional problem space.
2. For each particle, evaluate the desired optimization fitness function.
3. Compare particle's fitness value with particle's *pbest*. If current value is better than *pbest*, then set *pbest* value equal to the current value and the *pbest* position equal to the current position in D-dimensional space.
4. Compare the fitness evaluation value with the best swarm's fitness obtained so far. If current value is better than gbest, then reset gbest to the current particle's fitness value.
5. Change the velocity and position of the particle according to the equation (1) and (2) respectively.
6. Go back to step (2) until a termination criterion is met, usually a sufficiently good fitness or a specified number of generations.

3.1 Particle Position and Velocity Initialization and Limitation

For initialization of particle position, position vector x_{ij} is set to the random number from x_{min} to x_{max} .During a PSO run, position vector has no limitation bound. That is to say, the range [x_{min} , x_{max}] is valid only for initialization of position, which can assure that position sequence SP and operation sequence SO have the diversity, and then, schedule solutions decoded from the particle swarm have the diversity too. In the following Computation, x_{min} is set to 0, and x_{max} is set to 1.

For initialization of particle velocity, velocity vector v_{ij} is set to the random number from v_{min} to v_{max} . During a PSO run, velocity vector is limited to the range [v_{min} , v_{max}]. In the following computation, v_{min} is set to -1, and v_{max} is set to 1.

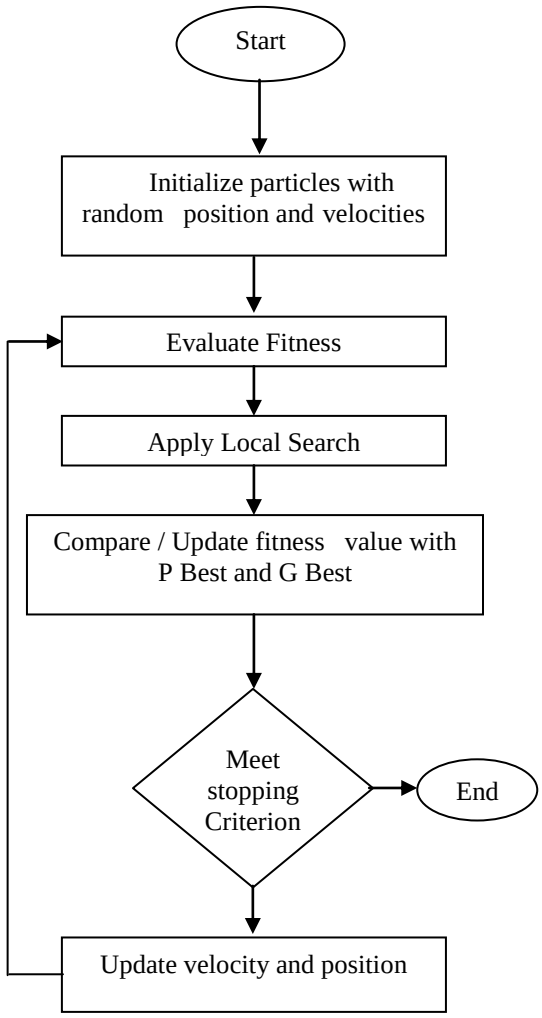

Figure 1: Particle Swarm Optimization Flow Chart

3.2 Particle Swarm Optimization Algorithm

Begin

Step 1. Initialization

Initialize parameters, including swarm size, maximum of generation, W_{max}, W_{min}, C1, C2;

Step 2. Assignment and scheduling

Generation=0;

Initialize particle's position and velocity stochastically;

Evaluate each particle's fitness;

```
Initialize gbest position with the particle with the lowest fitness in the swarm;

Initialize pbest position with a copy of particle it self;

While (the maximum of generation is not met)
  Do {
      generation = generation+1;

      Generate next swarm by equations;

      Evaluate swarm
      {
          Compute each particle's fitness;

          Find new gbest and pbest by comparison;

          Update gbest of the swarm and pbest of each particle;
      }
  }
```

Step 3. Output optimization results.

End.

3.3 Particle Representation (Encoding): Based on Operation and Particle Position Sequence

As PSO is used to optimize the problem, one key issue is the encoding, which is called particle representation in this paper. Suitable particle representation should importantly impact the optimization result and performance of PSO. In most applications of PSO, it is applied to the continuous optimization problems. In these optimization problems, particle position *xi* is directly denoted as the solution, which is continuous value. Velocity *vi* , acceleration constants C1 and C2 , and inertia weight W are also continuous constants. Because PSO model justly comprises addition, subtraction and multiplication operations, updated particle position and velocity are also continuous value.

Therefore, PSO completes the searching process in the continuous space that limits the use of PSO in the discrete space or combination optimization problem. However, job shop scheduling problem is a combination optimization problem, and its feasible solutions are the sequence of operations of all jobs.

The authors also solved the classical JSP problem using PSO. In case of JSP, PSO cannot directly employ the particle position as the solution. Certain particle representation should be employed, which can establish the mapping between the scheduling solution and the particle position, and the scheduling solution can be indirectly obtained through decoding of the particle representation. The paper employs the particle representation based on Operation- Particle Position Sequence (OPPS).

The feasible solution of JSP is the operation sequence of all jobs. For the particle position $xi=(x_{i1}, x_{i2}, \ldots x_{ij}, \ldots, x_{id})$, all position vectors x_{ij} (the total number is equal to d) also have a sequence (increasing sequence or decreasing sequence). So the operation sequence of all jobs and the sequence of the particle position vectors can be linked together, and the mapping is gained between the scheduling solution and the particle position.

Table 1: PSO results in comparison with the best known results of bench mark problems

Problem	m	n	Best known results	PSO results
ft06_csp:6*6	6	6	55	55
la01_csp:10*5	10	5	666	666
La02_csp:10*5	10	5	655	655
la03_csp:10*5	10	5	597	597
la04_csp:10*5	10	5	590	590
La06_csp: 15*5	15	5	926	926
La08_csp:15*5	15	5	863	863
La10_csp:15*5	15	5	951	951
La12_csp:20*5	20	5	1039	1039

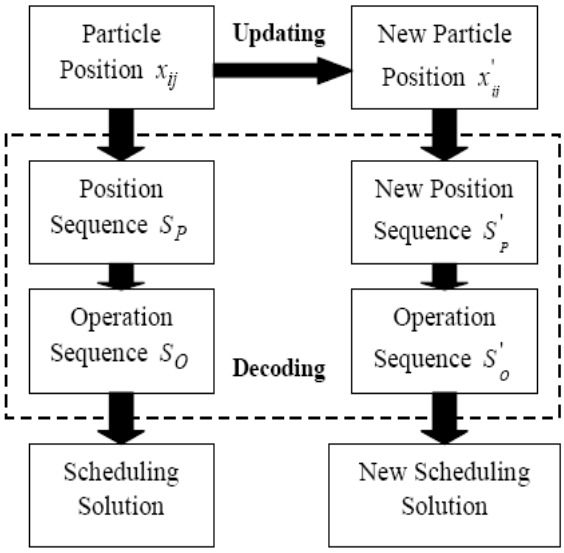

Figure 2: The mapping, decoding and updating of the particle

Particle swarm optimization algorithm is employed to solve the job shop scheduling problem with an objective of minimizing the maximum completion time of all the jobs. In the particle representation, the mapping between the particle and the scheduling solution is established through connecting the operation sequence of all the jobs with the particle position sequence. According to processing constraints of the problem, each operation in the operation sequence of all the jobs is assigned on each machine in turn to form the scheduling solution. The results of PSO based optimization heuristic are compared against the results of the benchmark problems in Table 1.

4.0 PSO for FJSP

The FJSP problem may be formulated as follows. Each instance of the problem is defined by set of jobs, a set of machines and a set of operations. Each job consists of a sequence of operations, each of which requires one machine out of a set of given machines. The problem is thus to determine both assignment and sequence of operations on all the machines that minimize the makespan subjected to the conditions: (i) the precedence of operations given by each job are to be respected. (ii) each machine can perform at most one operation at a time and (iii) the operations can not be interrupted.

Let:
J= {1, 2... n} denotes the set of jobs;
M= {1, 2… m} denotes the set of machines;
N= {0, 1, 2… n+1} denotes the set of operations, where 0 and n+1 represents the dummy start and finish operations, respectively.

The flexible job shop problem (FJSP) consists of a set J of n jobs that must be processed on a set M of m machines. Each job j consists of a sequence of n_j operations (routing) i.e. $O_{j,1}$, $O_{j,2}$, … $O_{j,,nj}$. The execution of each operation i of a job j ($O_{j,i}$) requires one machine out of a set of given machines called $M_{j,i} \subseteq M$.

4.1 Assumptions made in this work

- Each job is an entity: Although the job is composed of distinct operations, no two operations of the same job may be processed simultaneously. Thus we exclude from our discussion certain practical problems, e.g. those in which components are manufactured simultaneously prior to assembly into the finished product.
- No Pre-emption: Each operation once started, must be completed before another operation may be started on that machine.
- Each job has m distinct operations on one machine: we do not allow for the possibility that a job might require processing twice on the same machine.
- Setup times of machines and move time between operations are negligible.
- No machine may process more than one operation at a time.
- There is no randomness: in particular,
  - The number of jobs is known and fixed.
  - The number of machines is known and fixed.
  - The processing times are known and fixed.

4.2 Encoding

The most important issue in applying PSO successfully to FJSP is to develop an effective ‘problem mapping' and 'solution generation' mechanism. If these two mechanisms are devised successfully, it is possible to find good solutions for a given optimization problem in acceptable time. To find

a suitable mapping between problem solution and PSO particle, we first sequence the capable machines of an operation according to the increasing order of processing time. If one machine's processing time is equal to another, the lower order number of machine has priority. After this, we get different priority levels for all machines which process the same operation. Then the particle position can be generated stochastically according to the order of operations of different jobs.

The problem shown in Table 2 is to execute three jobs on four machines. Table 3 gives the order of priority of machines corresponding to each operation (priority 1>2>3>4).

Table 2: Flexible job shop

Job	Operation	Machine Number			
		M0	M1	M2	M3
J1	O11	2	3	4	1
	O12	3	1	8	2
J2	O21	1	4	1	2
	O22	5	3	2	9
	O23	3	1	1	4
J3	O31	7	6	3	5
	O32	4	5	6	2

Table 3: Priority order of machines.

Job	Operation	Priority Order			
		1	2	3	4
J1	O11	M4	M1	M2	M3
	O12	M2	M4	M1	M3
J2	O21	M1	M3	M4	M2
	O22	M3	M2	M1	M4
	O23	M2	M3	M1	M4
J3	O31	M3	M4	M2	M1
	O32	M4	M1	M2	M3

In general, initial particles' positions and initial particles' velocities in the swarm are generated at random. According to this approach to generate initial particles' positions, search space can be reduced and improves the search speed.

The PSO algorithm using the above encoding procedure is developed and tested for the FJSP and the results obtained after 200 iterations are as follows.

gbest initial sequence is 1 1 2 2 2 3 3
gbest machine sequence is 3 1 0 1 2 2 3
gbest ptime sequence is 1 1 1 3 1 3 2
gbest sequence is 2 2 3 1 1 2 3
Makespan (gbest) = 5

5. Conclusions

This paper discussed the Particle Swarm Optimization algorithm for FJSP problems. Although there is a huge literature on classical JSP, the FJSP does not have a rich literature. Therefore there is a need to develop effective approach for this complex problem. In this paper the authors have attempted JSP by using PSO and the experimental results show that Particle Swarm Optimization algorithm for JSP is very effective, and can find the best known solution for the cited benchmark instances. The preliminary results obtained by the implementation of PSO for FJSP in this paper are quite encouraging and motivating for the researchers to use PSO as a powerful tool in real time scheduling problems.

6. References

[1] Ajith Abraham, Hongbo Liu, Tae-Gyu Chang, "Variable Neighborhood Particle Swarm Optimization Algorithm".
[2] Chandrasekaran. S, Ponnambalam. S.G, Suresh. R.K, Vijayakumar. N, "An Application of Particle Swarm Optimization Algorithm to Permutation Flowshop Scheduling Problems to Minimize Makespan, Total Flowtime and Completion Time Variance".
[3] Kennedy J, Eberhart R C., "Particle Swarm Optimization", Proceedings of IEEE International Conference on Neutral Networks, Perth, Australia, 1995, pp. 1942-1948.
[4] Rahimi-Vahed A.R, Mirghorbani S.M, "A multi-objective particle swarm for a flow shop scheduling problem ", J Comb Optim (2007) 13:79–102 DOI 10.1007/s10878-006-9015-7.
[5] Sha D.Y, Cheng-yu Hsu, "A hybrid particle swarm optimization for job shop scheduling problem". Computers & Industrial Engineering 51 (2006) 791–808
[6] Weijun Xia, Zhiming Wu, "An effective hybrid optimization approach for multi- objective flexible job-shop scheduling problems". Computers & Industrial Engineering 48 (2005) 409–425.
[7] Zhixiong Liu, "Investigation of Particle Swarm Optimization for Job Shop Scheduling Problem".