J. Basic. Appl. Sci. Res., **1(12)3240-3244, 2011**
© 2011, TextRoad Publication

ISSN 2090-4304
Journal of Basic and Applied Scientific Research
www.textroad.com

A Particle Swarm Optimization Algorithm for Solving Flexible Job-Shop Scheduling Problem

Mohammad Akhshabi[1]* Mostafa Akhshabi[2], Javad Khalatbari[3]

[1*]Department of Industrial Engineering, Islamic Azad University Ramsar Branch, Ramsar, Iran
[2] Department of Computer Engineering, Islamic Azad University Ramsar Branch, Ramsar, Iran
[3] Department of Management, Islamic Azad University Ramsar Branch, Ramsar, Iran

ABSTRACT

In this paper we present a particle swarm optimization algorithm to solve the Flexible Job-shop Scheduling Problem with regard to to minimize the maximal completion time. In the view of its non-deterministic polynomial-time hard nature, the particle swarm optimization (PSO), which is inspired by the swarming or collaborative behavior of biological populations, is employed to solve minimize the maximal completion time. The performance of PSO is evaluated in comparison with CSA. Experimental results show that PSO significantly outperforms CSA.
KEY WORDS: Flexible Job-shop Scheduling Problem, particle swarm optimization algorithm, Makespan.

INTRODUCTION

Scheduling problems have a vital role in recent years due to the growing consumer demand for variety, reduced product life cycles, changing markets with global competition and rapid development of new technologies. The Job Shop Scheduling Problem (JSSP) is one of the most popular scheduling models existing in practice, which is among the hardest combinatorial optimization problems [1].Many approaches, such as, Simulated Annealing (SA) [2], Tabu Search (TS) [3], Genetic Algorithm (GA) [4], Ant Colony Optimization (ACO) [5], Neural Network (NN) [6], Evolutionary Algorithm (EA) [7] and other heuristic approach [8–10], have been successfully applied to JSSP.

In order to match today's market requirements, manufacturing systems need not only automated and flexible machines, but also flexible scheduling systems. The Flexible Job Shop Scheduling Problem extends JSSP by assuming that, for each given operation, it can be processed by any machine from a given set. Bruker and Schlie [11] were among the first to address this problem. The difficulties of FJSSP can be summarized as follows.

(1) Assignment of an operation to an appropriate machine;
(2) Sequencing the operations on each machine;
(3) A job can visit amachine more than once (called recirculation).

The flexible job shop problem (FJSP) is a generalization of the traditional job shop scheduling problem (JSP), in which it is desired to process operations on a machine chosen among a set of available ones. Therefore, the FJSP is more computationally difficult than the JSP, since it presents a further decision level besides the sequencing one, i.e., the job assignment. Determining the job assignment requires assigning each operation on a specific machine chosen among a set of machines. This problem is known to be strongly NP-hard even if each job has at most three operations and there are two machines [12]. The complexity of the FJSP suggests the adoption of heuristic methods producing reasonably good schedules in an acceptable execution time, instead of seeking for an exact solution. In recent years, the use of metaheuristics such as tabu search (TS) and genetic algorithms (GAs) has led to better results than classical dispatching or greedy heuristic algorithms [13–15].

In this paper, PSO algorithm according to the model is presented to tackle the FJSP. Experimental results on benchmark problems in the [16] show that FJSP can be solved by PSO effectively.

The rest of this paper is organized as follows. The details of the Flexible Job-shop Scheduling Problem are outlined in Section 2. Section 3 describes the PSO. And then in Section 4 the computational results are reported and discussed. This paper ends with conclusions in Section 5.

2. Flexible Job-shop Scheduling Problem

The n×m Flexible Job-shop Scheduling Problem (FJSP) can be formulated as follows:

***Corresponding Author:** Mohammad Akhshabi, Department of Industrial Engineering, Islamic Azad University Ramsar Branch, Ramsar, Iran. Email: m.akhshabi@toniau.ac.ir

There is a set of n jobs j= $\{j_1, j_2, \dots, j_n\}$ and a set of m machines M=$\{m_1, mj_2, \dots, m_m\}$ Each job J consists of a sequence of operations $o_{i1}, o_{i2}, \dots, o_{in_i}$ where n_i is the number of operations that j_i comprises. Each operation o_{ij} has to be processed by one machine out of a set of given machines $M_{ij} \in M$. the problem is thus to both determine an assignment and a sequence of the operations on the machines so that some criteria are attained. In this paper, the scheduling objective is to minimize the maximal completion time of all the operations, which is denoted by Makespan.

3. Particle swarm optimization

The PSO was first presented by Kennedy and Eberhart in 1995. In PSO, each potential solution is a point in the search space and may be regarded as a particle. Initially, a set of particles is randomly created and be moved through motion through the multi-dimensional problem space. In their movement, the particles have memory, and each particle regulates its position on the base of its own experience as well as the experience of a next particle by using the best position faced by itself and its neighbors. The previous best value is known as the *pbest* of the particle, and the best value of all the particles' *pbest* in the swarm is known as the *gbest*. In each repetition, the velocity and the position of each particle will be updated on the basis of to its *pbest* and *gbest*.

Imagine that the search space has *d*-dimensional, and the position of the *i*-th particle at the *t*-th iteration is shown by $X_i^t = (x_{i1}^t, x_{i2}^t, \dots, x_{iD}^t)$. The velocity of the *i*-th particle at the *t*-th repetition is suggested as $V_i^t = (v_{i1}^t, v_{i2}^t, \dots, v_{iD}^t)$. $P_i^t = (p_{i1}^t, p_{i2}^t, \dots, p_{iD}^t)$ is the best position of the *i*-th particle at the *t*-th iteration. The best position detected for all swarm is denoted as $P_g^t = (p_{g1}^t, p_{g2}^t, \dots, p_{gD}^t)$ at the *t*-th iteration. The velocity and position of particles are updated by Eqs. (1) and (2) in the standard PSO.

$$v_{id}^{t+1} = w \times v_{id}^{t+i} + c_1 \times rand() \times (p_{id}^t - x_{id}^t) + c_2 \times rand() \times (p_{gd}^t - x_{id}^t) \quad (1)$$

$$x_{id}^{t+1} = x_{id}^t + v_{id}^{t+1} \quad (2)$$

where X_i^t and V_i^t indicate the position and velocity of the *d*-th dimension of the *i*-th particle at the *t*-th iteration, respectively. w is the inertia weight represented to balance between the global and local search capabilities, and rand() is a random number creator with a uniform distribution over [0, 1]. c_1 and c_2 are the acceleration constants, which suggest the weighting of stochastic acceleration terms that force each particle toward *pbest* and *gbest*, respectively.

The updating formula forces particles moving toward compound vector of local and global optimal solutions. Therefore, opportunity for particles to get the optimal solution will be increased. Figure 1 shows the framework of the PSO for solving the flexible job-shop scheduling problem

3.1. Initial population

Each particle has its own position and velocity. The algorithm first begins by creating a population *P*, which is made of N_p particles. Each individual x_i also has d-dimensional parameter vectors. The index t denotes the iteration number of the algorithm. The population P is applied to show the positions of particle. The initial position population can be created randomly by using Eq. (5). m shows the number of identical parallel machines, and rand() is a random value between 0 and 1.

$$p(t) = \{x_1(t), x_2(t), \dots x_i(t), \dots, x_{N_p}(t)\} \quad (3)$$

$$x_i(t) = \{x_{1,i}(t), x_{2,i}(t), \dots, x_{j,i}(t), \dots, x_{D,i}(t)\} \quad (4)$$

$$x_{ij}(0) = rand() \times m + 1 \;\; ; \;\; i=1,2,\dots,N_p \;\; ; \;\; j=1,2,\dots,d \;\; ; \;\; t=1,2,\dots,T \quad (5)$$

A population *V*, which is composed of N_p individuals, suggests the velocity of each particle. It also has d-dimensional parameter vectors. The initial velocity population can be formed using Eq. (7). The velocity of each particle is equal to zeros. It means that all particles by velocity in the initial generation.

$$V(t) = \{v_1(t), v_2(t), \dots, v_i(t), \dots, v_{N_p}(t)\} \quad (6)$$

$$v_i(t) = \{v_{1,i}(t), v_{2,i}(t), \dots, v_{j,i}(t), \dots, v_{D,i}(t)\} \quad (7)$$

$V(0) = zeros\ (N_p, D)$

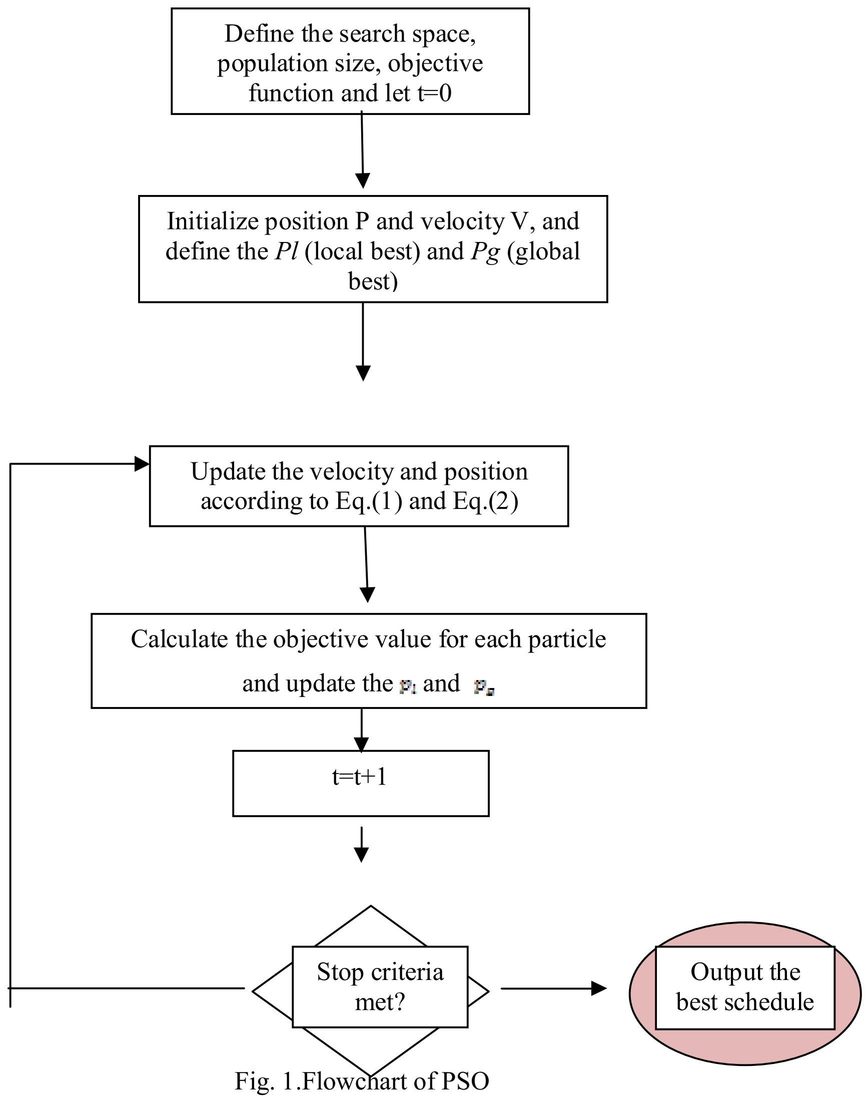
3.2. Selection

In GA, we use the roulette wheel selection technique. However, this operator does not warranty that each individual selected to enter the next generation is better than the last generation. In order to prevail this shortcoming, we adopt another selection strategy here. For each individual, we select the best individual to enter the next generation. In this article, we set the parameter values for MA selection as follows.

$L = 50\% \times N_p$; $\beta = 0.9$; $P_{hm} = 1$

3.3. Updating the velocity and the position

After performing the MA selection operation, the population will be done by PSO. According to Esq. (1) and (2), the velocity and position of the particles are updated. It is important to be mention that, on the basis of updating the velocity and the position, the main difference between PSO and PSO-based MA is the promotion of the current population. As we know, in PSO, the update plan permits each particle in the swarm to regulate its own velocity and position and will not be replaced by other potential particle. However, regarding PSO-based MA, because MA can yield a more diverse solution region, some particles in the swarm may be replaced by better solutions before performing PSO. Moreover, the local best positions will be promoted due to the contribution of MA. Therefore, it is possible for all particles in the swarm to benefit from their individual, as well as the swarm community, discoveries about the solution space.



Define the search space, population size, objective function and let t=0
Initialize position P and velocity V, and define the *Pl* (local best) and *Pg* (global best)
Update the velocity and position according to Eq.(1) and Eq.(2)
Calculate the objective value for each particle and update the p_l and p_g
t=t+1
Stop criteria met?
Output the best schedule

Fig. 1.Flowchart of PSO

4. EXPERIMENTAL RESULTS

In this paper, we mainly study three parameters of the two algorithms. They are the inertia weight w and the acceleration constants c1 and c2, respectively. The three parameters of PSO are determined the following, i.e., w=0.2, c1 = c2=2.0. To illustrate the effectiveness and performance of the hybrid algorithm proposed in this paper, it is implemented in MATLAB 7 on a laptop with Pentium IV Core 2 Duo 2.53 GHz CPU. The outputs of the CSPSO are compared with that achieved by [16] and it includes 12 test problems.

PSO run 20 times consecutively for each. Table 1 gives the results obtained by PSO and CSA in literatures. In the tables, C_{best} is the best solution obtained by Akhshabi. CPU is the required computing time in seconds. C* denotes the best solution obtained by PSO. Mean denotes the average solution that obtained by PSO. t denotes the computing time that obtained by PSO in seconds.

Table1. The result obtained by CSA and PSO

Problem	CSA			PSO	
	C_{best}	CPU	C*	Mean	t
9d	34	0.6	30	36	0.4
12d	42	0.6	42	47	0.4
16d	50	0.6	50	56	0.5
24d	60	0.8	54	63	0.5
28d	62	0.8	61	68	0.5
30d	70	0.8	64	73	0.7
252d	114	1	112	120	0.9
320d	201	2.48	187	209	1.74
500d	210	3.35	207	215	2.57
720d	300	5.78	286	321	4.13
1200d	318	7.62	307	338	5.87
2250d	426	9.16	419	439	8.12

From the Table 1, PSO gets better solutions in 10 of the 12 problems compared with CSA.

Results obtained by PSO are not very far from those obtained By Akhshabi, but the computational times have been reduced distinctly. The average solutions and the best solutions acquired by PSO are close too. It demonstrates at PSO can get solutions with high quality consistently.

In table (1) the results obtained from the Lingo8 software calculation and CSA calculation for 9 to 2250 dimensions have been compared

Table (2): Independent Samples Test

	Levene's Test for Equality of Variances		t-test for Equality of Means						
	F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	95% Confidence Interval of the Difference	
								Lower	upper
Equal variances assumed	6.292	.034	-1.73	16	.102	-.15150	.08740	-.33676	.03378
Equal variances not assumed			-2.14	15	.067	-.15150	.07165	-.30353	.00053

It seems that the mean of LINGO objective value equals the mean of CSA objective value. For exact statistical analysis, SPSS16 software was used and was calculated Independent Sample T Test, shown in table (2). As it is clear based on table(2), that significance value is 0.034 that means, equal variances not assumed and through examining we can consider that the significance value in second row i.e. 0.067 that shows the two means are equal.

5. Conclusions

In this paper, the PSO algorithm has been extended to deal with scheduling problem. The experiments prove that the PSO can be used to solve FJSP effectively.

The performance of the PSO has been calculated in comparison with the results obtained by the akhshabi[16], individually. The results clearly proved that the recommend PSO substantially outperforms CSA. Future research directions can be recommenced follows: (1) using the PSO- under an uncertain environment, including searching for an appropriate schedule where it is robust against some predefined interruptions, (2) using other famous meta-heuristics to compare the results with the ones reported by the PSO, (3) taking in to-consideration other objective functions at different practical constraints.

Acknowledgements

The author gratefully acknowledges the support provided by the Islamic Azad University Ramsar Branch for their kind support and cooperation during researched plan that this paper resulted from it.

REFERENCES

[1] M.R. Garey, D.S. Johnson, R. Sethi, The complexity of flow shop and job-shop scheduling, Mathematics of Operations Research 1 (2) (1996) 117–129.

[2] M. Kolonko, Some new results on Simulated Annealing applied to the Job Shop Scheduling Problem, European Journal of Operational Research 113 (1) (1999) 123–136.

[3] F. Pezzella, E. Merelli, A Tabu Search method guided by shifting bottleneck for the Job Shop Scheduling Problem, European Journal of Operational Research 120 (2) (2000) 297–310.

[4] J.F. Goncalves, et al., A hybrid genetic algorithm for the Job Shop Scheduling Problem, European Journal of Operational Research 167 (1) (2005) 77–95.

[5] K.L. Huang, C.J. Liao, Ant colony optimization combined with taboo search for the job shop scheduling problem, Computers and Operations Research 35 (4) (2008) 1030–1046.

[6] D.J. Fonseca, D. Navaresse, Artificial neural networks for job shop simulation, Advanced Engineering Informatics 16 (4) (2002) 241–246.

[7] I.T. Tanev, T. Uozumi, Y. Morotome, Hybrid evolutionary algorithm-based realworld Flexible Job Shop Scheduling Problem: application service provider approach, Applied Soft Computing 5 (1) (2004) 87–100.

[8] H. Chen, P.B. Luh, An alternative framework to Lagrangian relaxation approach for Job Shop Scheduling, European Journal of Operational Research 149 (3) (2003) 499–512.

[9] W.Q. Huanh, A.H. Yin, An improved shifting bottleneck procedure for the Job Shop Scheduling Problem, Computers & Operations Research 31 (12) (2004) 2093–2110.

[10] K. Jansen, M. Mastrolilli, R. Solis-Oba, Approximation schemes for Job Shop Scheduling Problems with controllable processing times, European Journal of Operational Research 167 (2) (2005) 297–319.

[11] P. Bruker, R. Schlie, Job-shop scheduling with multi-purpose machines, Computing 45 (4) (1990) 369–375.

[12] Garey MR, Johnson DS, Sethi R. The complexity of flow shop and job shop scheduling. Mathematics of Operations Research 1976;1:117–29.

[13] Mastrolilli M, Gambardella LM. Effective neighbourhood functions for the flexible job shop problem. Journal of Scheduling 2000;3:3–20.

[14] Pezzella F,Morganti G, Ciaschetti G. A genetic algorithm for the flexible job-shop scheduling problem. Computers & Operations Research 2008;35(10):3202–12.

[15] Gao J, Sun L, Gen M. A hybrid genetic and variable neighborhood descent algorithm for flexible job shop scheduling problems. Computers & Operations Research 2008;35(9):2892–907.

[16] Akhshabi M, Akhshabi M, Khalatbari J. Solving flexible job-shop scheduling problem using clonal selection algorithm. Indian Journal of Science and Technology 2011:10(4): 1248_51