Investigation of Particle Swarm Optimization for Job Shop Scheduling Problem*

Zhixiong Liu
College of Machinery and Automation, Wuhan University of Science and Technology, 430081, Heping Road 947, Wuhan, Hubei Province, China
Lzx_brad@126.com

Abstract

Job shop scheduling problem has stronger processing constraints, and it is a kind of well-known combination optimization problem. Particle swarm optimization algorithm is employed to solve the job shop scheduling problem, and the objective is minimizing the maximum completion time of all the jobs. The particle representation based on operation-particle position sequence is proposed. 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 particle representation can ensure that the scheduling solutions decoded are feasible and can follow the particle swam optimization algorithm model. The computational results show that particle swarm optimization algorithm can effectively solve the job shop scheduling problem.

1. Introduction

Scheduling is concerned with allocating limited resources to tasks to optimize some performance criterion, such as completion time or production cost. Job shop scheduling (JSP) is classic and popular problem in scheduling area. In general, the job shop scheduling problem can be described as follows: Given n jobs, each must be processed on m machines. Each job consists of a sequence of operations, which must be executed in a specified order. Each operation has to be performed on a given machine for a given time. A schedule is an allocation of the operations to time intervals on all machines. The objective to optimize is to find the schedule that the maximum completion time of all the jobs (makespan) is minimal. The constrains are as following: 1) the operation precedence is respected for every job, 2) each machine can process at most one operation at a time, and 3) an operation can not be interrupted if it initiates processing on a given machine.

Because job shop scheduling problem is the combination optimization problem and belongs to the class of NP-complete problems [1], the use of mathematical methods, such as integer programming or mixed-integer programming, have been limited to solve the small problems. In recent years, some heuristics approaches have been developed, which is a quite good alternative, such as simulated annealing (SA) [2], taboo search (TS) [3], and genetic algorithms (GA) [4]. Heuristics approach can often find high-quality solutions in a reasonable computation time instead of finding optimal solution. So many practitioners give much attention to heuristics approaches.

Particle swarm optimization (PSO) is one of the latest evolutionary optimization methods inspired by nature which include evolutionary strategy (ES), evolutionary programming (EP), genetic programming (GP) and GA. Since PSO was first introduced to optimize various continuous nonlinear functions by Kennedy and Eberhart [5], [6], [7], it has been successfully applied to a wide range of applications such as power and voltage control, neural network training, task assignment and ordering problem [8], etc. Although the applications of PSO on combination optimization problems are still limited, PSO has its merit in the easy implementation and computation efficiency. In scheduling area, PSO has been applied to the permutation flow shop scheduling problem [9] and the single-machine scheduling problem [10], [11].

In the paper, PSO is employed to solve the job shop scheduling problem with the minimal makespan. The remainder of the paper is organized as follows. Section 2 describes PSO algorithm with linearly decreasing inertia weight. Section 3 presents that how PSO is applied to the job shop scheduling problem. In section 4, PSO is used to solve some benchmark instances for the job shop scheduling problem, and compared with genetic algorithm. Some concluding remarks are made in section 5.

*The open research project supported by the Project Fund of the Hubei Province Key Laboratory of Mechanical Transmission and Manufacturing Engineering Wuhan University of Science and Technology (No：2005A17).

0-7695-2875-9/07 $25.00 © 2007 IEEE

IEEE COMPUTER SOCIETY

2. Particle Swarm Optimization Algorithm

PSO algorithm is motivated from the simulation of social behavior of bird flocking, and it is the optimization method based on population (named swarm in PSO). PSO is distinctly different from other evolutionary-type methods (such as genetic algorithm) in a way that it does not use the filtering operation (such as crossover and /or mutation) and the individuals of the entire swarm 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. In PSO, each single solution is "bird" in the search space, which is called "particle". All particles have fitness values, which are evaluated by the fitness (or objective) function to be optimized. Every particle in the swarm is described by position and velocity. In general, particle position represents the solution of optimization problem, and velocity represents the search distance and direction, which guides the flying of the particle. All particles fly through the problem space by following the current best particles. The current best particles consist of the local best particle and the global best particle. The local best particle represents the best solution (position) the particle has achieved so far, and the global best particle represents the best value (position) obtained by the swarm so far. Up to the present, researchers have explored several models about PSO [8], and this paper employs the PSO model with linearly decreasing inertia weight.

Suppose that the search space is D-dimensional, and then the i-th particle of the swarm can be represented by a D-dimensional vector, $x_i = (x_{i1}, x_{i2}, \cdots, x_{ij}, \ldots, x_{id})$. The velocity of the particle can be represented by another D-dimensional vector, $v_i = (v_{i1}, v_{i2}, \cdots v_{ij}, \cdots, v_{id})$. The local best particle is denoted as $p_i = (p_{i1}, p_{i2}, \cdots p_{ij}, \cdots, p_{id})$, and the global best particle is denoted as $g = (g_1, g_2, \cdots g_j, \cdots, g_d)$. The particle updates its velocity and position by means of equation (1a) and (1b) as follows.

$$v_{ij} = \omega v_{ij} + c_1 random()(p_{ij} - x_{ij}) + c_2 random()(g_j - x_{ij}) \quad (1a)$$

$$x_{ij} = x_{ij} + v_{ij} \quad (1b)$$

In equation (1a), ω denotes inertia weight, and it is an important parameter to search ability of PSO algorithm. A large inertia weight facilitates searching new area while a small inertia weight facilitates fine-searching in the current search area. Suitable selection of the inertia weight can provide a balance between the global exploration and local exploitation abilities of the swarm, and results in fewer iterations on average to find a sufficiently optimal solution. Therefore, considering by linearly decreasing the inertia weight from a relatively large value (the initial value) to a relatively small value (the final value) with the iterations during a run, PSO tends to have more global search ability at the beginning of the run while having more local search ability near the end of the run. In this paper, the initial value of the inertia weight is set to 0.9 at the beginning of the run, and the final value is set to 0.4 at the end of the run.

The acceleration constants c_1 and c_2 in equation (1a) represent the weight of the stochastic acceleration terms that pull each particle toward p_i and the g positions. Thus, adjustment of these constants changes the amount of "tension" in the system. Low values allow particles to roam far from target regions before being tugged back, while high values result in abrupt movement towards, or past, target regions. Popularly, both c_1 and c_2 are set to 2.0. In equation (1a), *random*() is random function with range [0, 1].

As same as other evolutionary optimization methods, PSO based on swarm intelligence employs the search strategy based on population. However, most important differently, PSO has an obvious mathematic formula, through which all particles in entire swarm are updated. Therefore, PSO is easy to be manipulated and implemented.

3. PSO for Job Shop Scheduling Problem

3.1. 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 x_i is directly denoted as the solution, which is continuous value. Velocity v_i, acceleration constants c_1 and c_2, and inertia weight ω are also continuous constants. Because PSO model (equation (1a) and (2b)) 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. Applied to solve the 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

IEEE COMPUTER SOCIETY

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 $x_i = (x_{i1}, x_{i2}, \cdots x_{ij}, \cdots, x_{id})$, 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.

For JSP consisting of n jobs and m machines, every job must be processed on m machines. That is to say, every job has m operations while every job has possible different operation sequence. All jobs have totally l operations, $l = n * m$. Based on above consideration, define that the search space of PSO is l-dimensional, and the particle representation for JSP is denoted by table 1 as follows.

Table 1. Particle representation for JSP

Operation	1	1	⋯	k
Position x_{ij}	x_{i1}	x_{i2}	⋯	$x_{i((k-1)*m+1)}$
Operation	⋯	k	⋯	n
Position x_{ij}	•••	$x_{i(k*m)}$	•••	x_{il}

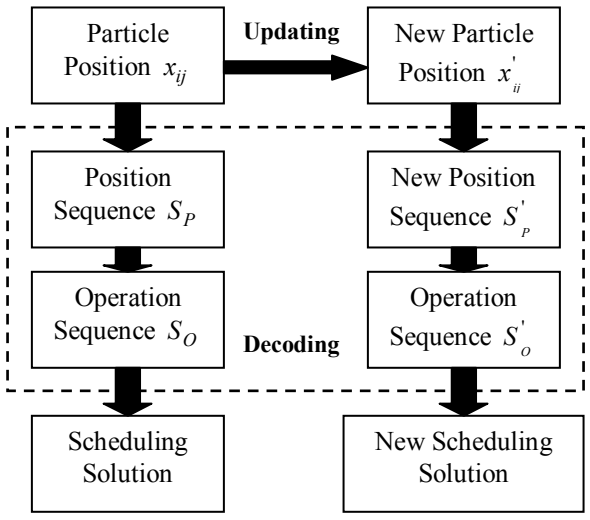

Figure 1. The mapping, decoding and updating of the partilce representation based on OPPS

In table 1, the length of particle is l. The first line in table 1 denotes all operations of all the jobs. Every operation of k-th job is numbered k, and there are m same number k in the first line. The second line denotes the i-th particle position. The sequence of all operations in first line will correspondingly change if all particle position vectors in second line are sequenced. So an operation sequence of all the jobs will be obtained. And then, according to the constraint of JSP, each operation in the operation sequence of all the jobs is assigned on each machine in turn and a scheduling solution of JSP will be done. If the operation sequence of all the jobs is denoted as S_O, and the sequence of all the particle position vectors is denoted as S_P, the mapping, decoding and updating of the particle representation based on OPPS is described by figure 1 as above.

In figure 1, while the particle position x_{ij} is updated to x'_{ij}, position sequence S_P is updated to S'_P, and then, operation sequence is changed from S_O to S'_o. At last, old scheduling solution is replaced by the new scheduling solution. During the computation of a PSO run, original sequences of operation (the first line) and particle position (the second line) in table 1 cannot be changed. With the iterations, while values of the particle position vector x_{ij} are continuously replaced, original values of the operation do not be changed. In this way, updating of the particle can follow the PSO model. The global best particle and the local best particle are selected through evaluating the maximal completion time of all the jobs (makespan) for the scheduling solution.

Table 2. The processing time and constraints of JSP (J1, J2 and J3 denote 3 jobs. M1, M2 and M3 denote 3 machines)

	Job	Processing Sequence		
		1	2	3
Processing Time	J1	3	3	2
	J2	1	5	3
	J3	3	2	3
Sequence of Machines	J1	M1	M2	M3
	J2	M1	M3	M2
	J3	M2	M1	M3

Table 3. The i-th particle representation

Operation	1	1	1	2	2
Position x_{ij}	1.23	2.12	1.31	3.08	1.18
Operation	2	3	3	3	
Position x_{ij}	2.45	4.87	3.67	6.75	

For example, suppose that JSP consists of 3 jobs and 3 machines. Table 2 describes the processing time and constraint. Suppose table 3 describes the i-th particle of the swarm.

IEEE COMPUTER SOCIETY

In table 3, after increasingly sequencing the position vector x_{ij}, an increasing sequence S_O of all operations of 3 jobs, S_O = [2,1,1,1,2,2,3,3,3], can be gained. Above all, the first number in the sequence S_O is 2, and according to the constraint in table 2, the 1-th operation of 2-th job is processed on machine M1. In succession, because the second number in the sequence S_O is 1, the 1-th operation of 1-th job is processed on machine M1. And then, the third number in the sequence S_O is 1, so the 2-th operation of 1-th job is processed on machine M2. So does it in turn. Finally, all operations of 3 jobs are assigned on 3 machines. Thus, a scheduling solution is obtained.

It is obvious that each scheduling solution from particle presentation based on OPPS and decoding should be feasible. On the other hand, every particle has an only (increasing or decreasing) sequence, and correspondingly, an only operation sequence and an only scheduling solution are gained after decoding. Moreover, every particle is initialized randomly, so PSO based on swarm can produce more different scheduling solutions, which poses that parallel search is started with different initial scheduling solutions.

3.2. Particle Position and Velocity Initializati-on 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 S_P and operation sequence S_O 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 2.

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 -2, and $v_{\max}$ is set to 2.

In general, the process for implementing the PSO for JSP is as follows.

① Initialize a swarm of particles positions and velocities.

② Obtain the scheduling solutions through decoding the particle representation.

③ Find the global best particle and the local best particle through evaluating the maximal completion time of all jobs (makespan) for scheduling solutions.

④ Updating the particles positions and velocities through equation (1a) and (1b).

Loop to step ② until a maximum number of iterations is met.

4. Experimental Results

To illustrate the performance of introduced PSO for JSP in this paper, eleven benchmark instances with different sizes have been selected to compute, which are cited from OR-library [12]. PSO for JSP is programmed in Visual Basic and run on Intel Pentium 1.7G with 256M rams. Moreover, swarm size is set to 40 and maximum of iterative generations is set to 300. Each instance is randomly performed 20 times. Computational results for PSO are shown in table 4 and compared with GA. The solution quality is measured with the minimum makespan of PSO and the percent relative increase in makspan with respect to the best known solutions from OR-library. In table 4, Best Solution denotes the minimum makespan and Mean Deviation denotes the average percent relative increase in makespan with respect to the best known solution after 20 respective runs.

Table 4.Computational Results for PSO and GA for JSP

Probl-em	Best known Soluti-on	GA		PSO	
		Best Soluti-on	Mean Deviati-on (%)	Best Soluti-on	Mean Deviati-on (%)
FT06	55	55	3.273	55	0
FT10	930	997	11.871	945	11.46
FT20	1165	1247	12.403	1169	11.55
LA01	666	666	3.048	666	0.345
LA06	926	926	0	926	0
LA11	1222	1222	0	1222	0
LA16	945	979	6.032	949	5.13
LA21	1046	1156	14.56	1140	12.35
LA26	1218	1328	12.783	1281	11.99
LA31	1784	1836	14.826	1812	12.91
LA36	1268	1384	13.908	1321	13.17

From above computational results, for some benchmark instances, such as FT06, LA01, LA06 and LA11, PSO for JSP can find the scheduling solution as same as the best known solution. With regard to the other benchmark instances, PSO for JSP can also find a better hypo-optimal scheduling solution than GA. On the other hand, to all instances, PSO is better than GA in Mean Deviation. However, implement and manipulation of PSO are easier than GA.

5. Conclusion

In above description, the particle swarm optimization algorithm for JSP has been discussed. A kind of particle

IEEE COMPUTER SOCIETY

representation and decoding approach for JSP has been proposed. Experimental results show that particle swarm optimization algorithm for JSP is very effective, and can find the best known solution or the hypo-optimal solution for the cited benchmark instances. As the same time, computational results indicate that PSO is an attractive alternative for solving the job shop scheduling problem. Adjusting the particle representation for JSP less, PSO can also be employed to solve the other scheduling problem and combination optimization problem, and then the author has been processing the work. In the future, to improve the performance of PSO for JSP, the following work would be done. Firstly, the parameters of PSO for JSP would be adjusted, such as inertia weight, size of swarm, and size of particle position and velocity initialization range, etc. Secondly, synthetic method of PSO and the other optimization approaches would be employed. Finally, the other PSO models would be analyzed.

References

[1] B.J. Lageweg, E.L. Lawler, J.K. Lenstra, "A.H.G. Rinnooy Kan: Computer-Aided Complexity Classification of Combinatorial Problems". Comm. ACM 25,1982, pp. 817-822.

[2] Van Laarhoven PJM, Aarts EHL, Lenstra JK, "Job Shop Scheduling by Simulated Annealing", Annals of Operations Research, Vol.40, 1992, pp. 113-125.

[3] Dell Amico M., Trubian M., "Applying Taboo Search To the Job Shop Scheduling Problem", Annals of Operations Research, Vol.41, 1993, pp. 231-252.

[4] Ana Madureira, Carlos Ramos, Silvio do Carmo Silva, "A Genetic Approach for Dynamic Job-Shop Scheduling Problems", Proceedings of 4th Metaheuristics International Conference, July 16-20, Porto, Portugal , 2001, pp. 41-45.

[5] Kennedy J, Eberhart R C., "Particle Swarm Optimization", Proceedings of IEEE International Conference on Neutral Networks, Perth, Australia, 1995, pp. 1942-1948.

[6] Eberhart R，Kennedy J., "A New Optimizer Using Particle Swarm Theory", The sixth International Symposium on Micro Machine and Human Science , 1995, pp. 39-43.

[7] Shi Y H, Eberhart R C., "A Modified Particle Swarm Optimizer", IEEE International Conference on Evolutionary Computation, May 4-9, Anchorage, Alaska , 1998, pp. 69-73.

[8] Eberhart R C, Shi Y H., "Particle Swarm Optimization: Development, Applications and Resources", Proceedings of the Congress on Evolutionary Computation, Seoul, Korea, 2001, pp. 81-86.

[9] M. Fatih Tasgetiren, Mehmet Sevkli, Yun-Chia Liang, and Gunes Gencyilmaz, "Particle Swarm Optimization Algorithm for Permutation Flowshop Sequencing Problem", Proceedings of the 4th International Workshop on Ant Colony Optimization and Swarm Intelligence LNCS 3172, September 5-8, Brussels, Belgium, 2004, pp. 382-390.

[10] M. Fatih Tasgetiren, Mehmet Sevkli, Yun-Chia Liang, and Gunes Gencyilmaz, "Particle Swarm Optimization Algorithm for Single Machine Total Weighted Tardiness Problem", Proceedings of the Congress on Evolutionary Computation, June 20-23, Portland, Oregon, 2004, pp. 1412-1419.

[11] Leticia Cagnina, Susana Esquivel, Raul Gallard, "Particle Swarm Optimization for Sequencing Problems: A Case Study", Proceedings of the Congress on Evolutionary Computation, Oregon, Portland, 2004, pp. 536-540.

[12] OR Library, http://mscmga.ms.ic.ac.uk/

0-7695-2875-9/07 $25.00 © 2007 IEEE

IEEE COMPUTER SOCIETY