International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Index Copernicus Value (2015): 78.96 | Impact Factor (2015): 6.391

Solving Job-Shop Scheduling Problem Using a Developed Particle Swarm Optimization Algorithm

Wathiq N. Abdullah

Department of Computer Science, University of Baghdad, College of Education for Pure Science- Ibn AlHaitham

Abstract: ***This paper aims to provide an efficient optimization algorithm to solve the problem of job-shop scheduling. PSO has been widely studied in numerous applications for its worthy global searching ability. A new modified of particle swarm optimization algorithm is adopted, which is an extension of the standard particle swarm optimization algorithm (PSO) in order to get a better performance. The improvement technique used is the overcome of the idle particle position. The system used the developed PSO as a search method for an optimal schedule. The experimental results show that the developed PSO performs better than the original PSO.***

Keywords: Job-Shop, Scheduling, Particle Swarm, Optimization

1. Introduction

Manufacturing processes have become increasingly complex, and so have their planning and control. The market requires high product variety, high quality, short lead times, and an accurate delivery performance [1].

Scheduling problems are usually approached with a mixture of search techniques and heuristics. These problems are likely to be uncontrollable and cannot be solved by combinatorial search methods. Furthermore, they involve a competition for limited resources; as a result, they are complex by various restrictions [2].

In the past, each machine had its own operator, its own tools, and so on. Nowadays, tools are more versatile, hence can be used by various different machining centers. These tools are, however, at the same time more expensive. Therefore, a company often decides to cut its tool investments. Similarly, operators are expensive, but often they only need to tend a machine during part of the operations, e.g., to position a job on the machine. This enables the operators to tend more than one machine; therefore, the number of operators is often smaller than the number of machines.

Such an operator and tool sharing clearly increases the interdependency of the machines, because a unique tool can only be used by one machine at a time and an operator can only tend one machine at a time. The planning and control must take these dependencies into account and the machines must be planned and controlled simultaneously to realize short lead times and a good delivery performance [3].

Local search approaches can find the solutions, but the worth of a solution and computational time be influenced by to a unlimited degree on suitable initial populations [4]. As a result of the initiation of computation techniques, metaheuristics can be used to solve problems in less time so that the limitation of computational complexity can be fixed by metaheuristic applications [5] This paper tries to adopt a metaheuristic technique, Particle Swarm Optimization algorithm to solve the problem of job-shop scheduling.

The scheduling problems are complex for the following reasons: (i) Scheduling is a practicality problem. The final solution must achieve all the constraints of the problem. Also, the optimization of an evaluation function should be satisfied, altering certain criteria as cost, delay, inventory time, process time, etc. (ii) Several scheduling problems require many restrictions because of the unavailability of resources, due dates, etc. [6]

The remainder of this paper is organized as follows: the formulation of job-shop scheduling problem, the particle swarm optimization algorithm of solving job-shop scheduling problem, and finally, the results of the system and some conclusions' remarks are given.

2. Job-Shop Scheduling Problem Formulation

The JSP is a scheduling problem that assumes M various machines and N various jobs. Each job consists of Q operations and each operation needs a different machine. The jobs' operations are handled in a fixed processing order [7] which specifies the precedence restrictions [8]. The operations of a job are totally ordered so that no operation of a job can start before the completion of its predecessor [9].

Scheduling systems typically rely on priority rules, which have therefore become the subject of intense study [10].

Two kinds of constraints need to be considered for the Job-Shop Problem [11] as follows:

1) The constraint of operation precedence for a specific job:
Let cij denote the time to complete the job i on machine j and let tij denote the time to process the job i on machine j. For a job i, if the processing on machine h precedes that on machine j, the following constraint should be satisfied:

$$cij - tij >= cih \quad \ldots\ldots\ldots (1)$$

2)The constraint of operation un-overlapping for a specific machine:

For two jobs i and j, both need to be processed on machine k. If job I comes before job j, we need the following

constraint [7]:

$$cjk - cik >= tjk \ldots\ldots\ldots \quad (2)$$

The goal of the Job-Shop Problem is to choose for each operation a suitable machine and a starting time so that the maximum completion time Cmax (the makespan) is minimized [12, 13, 14].

3. Particle Swarm Optimization

The particle swarm principle was inspired by the model of social behaviors. The Particle Swarm Optimization (PSO) algorithm creates the simple manner procedures for each particle, recalls the best position of the particles, and shares the information among particles. However, this algorithm accomplishes the optimization through cooperation and competition between the populations' individuals [15].
The Particle Swarm Optimization system is a member of the extensive group of swarm intelligence approaches, to solve the problems of optimization [16]. It was firstly suggested by J. Kennedy [17]. The basic PSO model consists of a swarm of particles, which are a random initialization of population of candidate solutions.

The modified PSO algorithm is described as follows.
STEP 1: Create randomly an initial particle swarm, by setting the initial position and the initial velocity of each particle;
STEP 2: Compute fitness value for each particle;
STEP 3: Compare each particle fitness value and its best position fitness value if better, update the position;
STEP 4: Compare each particle best position and the best position of particle swarm, if better, update the best position;
STEP 5: modify the position and velocity;
STEP 6: Perform the improvement process: avoiding the idle position;
STEP 7: Checking of the termination criteria are satisfied (get good sufficient position or maximum number of iterations are reached), then end; otherwise, go to 2. [15]

4. PSO for Job-Shop Scheduling Problem

The job-shop scheduling problem is a problem of M machines and N jobs. Each job consists of a series of processes. All the processes of each job are treated in a fixed processing order.

Suppose that the available set of total operations of N jobs be OP, the available set of machines is M, and the indices of the operation op be i and k, then we have:
OP = {op ik | i = 1, 2, . . ., N and k = 1, 2, . . ., M } ... (3)

Where i is the job number, and k indicates the operation number of jobi. The ik indices of the operation op are encoded as a sequence number (index).

Each particle can be represented by two fields: the first field (Mach) consists of M-bit string (M is the number of machines), where each bit corresponds to one machine. If the operation is to be processed on a particular machine, the corresponding bit assumes value (1), otherwise it is (0). The second field of a gene is the completion time of the operation (C).

4.1 Generation of Initial Swarm

The initialization procedure produces swarm_size particles- where pop_size denotes the swarm size- by setting (1) at random position in each (mach) field of a particle and filling the remainder bits of the field by (0's) until the whole particle is filled and repeat the same procedure until the initial swarm is completed.

4.2 Fitness Evaluation

The evaluation of a particle's fitness involves finding the total completion time (TCT) of the operations on each machine. On each machine, the total completion time can be calculated by summing the time required to process each operation on a specific machine. The total completion time having the maximum value is the particle's fitness value. The fitness function can be evaluated as follows:

Particle's Fitness = Maximum of (TCT_R) = Max (TCT_R)

$$= \left(\sum_{i=1}^{n} T_{P_i} \times Particle[P_i[b_R]]\right) \quad \ldots \quad (4)$$

Where:
R is the machine number: R is between (1) and total number of machines,
n is the total number of operations,
Tp_i is time required to perform the operation number *i* , and $Particle[P_i[b_R]]$ is a binary value located in the particle at operation number (*i*) at bit number (*R*).

4.3. PSO Improvement Approach: Avoiding Idle Position

Assume that "R" is a number used to decide whether a particle's position is ignored or not. The initial value of R for each particle is zero. If the particle's position was not enhanced over the execution of the algorithm, then R will be increased by 1; else R=0. If the position can never be enhanced, then the position will be is ignored and a it will be substituted by a new position which is produced by the equation (5):

New position = w * pg + h * (pg –Old position) ….. (5)
where:
w: inertia weight,
pg: is particle's global best fitness value, and
h: is a random number in the range [0,1].

5. Experimental Results

Several various experimental cases have been taken. Each case differs from each other by swarm-size and maximum number of iteration.

The PSO parameters values that used in the system experiments are given in Table (1) and the values of parameters used in each instance are given in Table (2).

Different instances are implemented by the system which are shown in Table () through Table (). Each instance has different number of machines, different number of jobs, different number of operations per job. The time required to perform operations of any job are generated randomly.
In instances tables, the *(M, T)* field means that an operation of a job should go to machine (*M*) for (*T*) units of time. At the end of each instance table the resultant makespan is given.

Table 1: The PSO Parameters.

Parameter	*Value*
Self-confidence (C_1)	1.4
Swarm confidence (C_2)	1.4
Inertia weight (W)	0.8
The maximum velocity (V_{max})	25
Number of particles in the swarm	30-100
The maximum number of iterations	50-100

Table 2: Problem Parameters Used in the Instances.

	Instance 1	Instance 2	Instance 3	Instance 4
Swarm_Size	30	30	40	100
Max_Iteration	50	100	50	100
No. of Machines	6	6	10	6
No. of Jobs	4	6	8	10

Table 3: Problem Instance 1

	(M , T)	(M , T)	(M , T)	(M , T)	(M , T)	(M , T)
Job 1	6,5	3,8	5,12	2,7	1,8	4,5
Job 2	1,6	6,4	4,8	5,10	3,12	2,9
Job 3	3,6	2,12	5,8	1,4	4,1	6,6
Job 4	6,5	5,3	3,1	4,12	2,5	2,14
The Makespan is 36						

Table 4: Problem Instance 2

	(M , T)	(M , T)	(M , T)	(M , T)	(M , T)	(M , T)
Job 1	7,2	5,10	2,4	1,6	4,8	3,10
Job 2	6,11	8,5	4,5	1,6	6,14	3,6
Job 3	2,4	7,7	5,10	4,9	1,11	8,5
Job 4	8,2	3,10	4,6	7,10	6,2	5,9
Job 5	3,8	5,8	4,10	8,10	2,13	7,6
Job 6	2,7	3,10	8,4	5,10	4,6	6,8
The Makespan is 40						

Table 5: Problem Instance 3

	(M , T)	(M , T)	(M , T)	(M , T)	(M , T)	(M , T)	(M , T)	(M , T)	(M , T)	(M , T)
Job 1	2,12	10,5	4,6	9,1	3,10	7,8	1,7	5,6	2,6	9,4
Job 2	8,5	6,4	10,14	3,8	4,6	9,3	1,10	2,10	7,7	5,11
Job 3	5,7	1,7	10,6	2,9	9,10	3,12	6,10	4,3	7,5	8,4
Job 4	1,9	9,4	5,8	6,6	1,5	6,12	3,9	8,10	4,10	2,3
Job 5	10,4	3,10	4,12	6,5	7,6	2,8	3,7	9,3	5,8	7,5
Job 6	7,10	6,5	1,7	1,12	3,14	2,6	4,3	4,9	5,9	6,8
Job 7	9,6	2,8	3,10	8,11	7,10	6,5	1,4	4,7	6,6	5,10
Job 8	2,12	10,5	4,6	9,1	3,10	7,8	1,7	5,6	2,6	9,4
The makespan is 65										

Table 6: Problem Instance 4

	(M, T)	(M, T)	(M, T)	(M, T)	(M, T)	(M, T)
Job 1	1,7	6,10	5,14	2,2	4,8	6,1
Job 2	6,6	1,8	5,6	3,12	4,14	2,10
Job 3	3,10	5,4	4,7	1,4	2,14	6,12
Job 4	6,4	4,7	1,4	2,12	3,10	5,1
Job 5	1,5	2,14	5,10	6,1	5,2	3,7
Job 6	5,5	4,5	3,14	6,10	1,7	2,8
Job 7	2,7	6,14	5,8	1,11	3,9	4,5
Job 8	5,11	6,4	4,12	3,8	1,6	2,6
Job 9	4,10	3,6	6,4	5,7	2,8	1,4
Job 10	3,9	6,7	5,11	3,14	4,4	1,12
The Makespan is 96						

6. Conclusions

This paper presents a developed particle swarm optimization algorithm for Job Shop Scheduling Problem.

A number of machines are included in the system. Each machine can process only one job at a time. Each job contains a number of operations. The rule of a priority is adopted to construct the schedule. The system is tested with several different problem instances which are differ from each other by the number of jobs, the number of operations required to be processed in each job, the number of machines, and the values of PSO parameters. The problem is to find a schedule having a minimum total time often called the *makespan* of a schedule. The development of the PSO algorithm is to avoiding the existence of idle particle position. The experimental results show that the developed PSO algorithm gives near-optimal schedules on all instances.

References

[1] J.D. Blackburn, *Time-Based Competition, The Next Battle Ground in American Manufacturing*,Richard D. Irwin, Homewood, IL, 1991.
[2] Negenvitsky, M., Artificial Intelligence: A Guide to Intelligent Systems. 2nd ed, Addison-Wesley, Harlow, England, 2005.
[3] Schutten, J.M.J., Practical job shop scheduling, *2014*.
[4] Liu, J.; Reeves, C. Constructive and composite heuristic solutions to the P//#C_i scheduling problem. European Journal of Operational Research 132, 439–452, **2001**.
[5] Laxmi A. Bewoor, V. Chandra Prakash and Sagar U. Sapkal , " Evolutionary Hybrid Particle Swarm Optimization Algorithm for Solving NP-Hard No-Wait Flow Shop Scheduling Problems", 2017.
[6] Garrido, A., Salido, M. A., Barber, F., and Lopez, M. A., *Heuristic Methods for Solving Job-Shop Scheduling Problems*,Spain, 1998.
[7] Liu, T.-K., Tsai, J.-T., and Chou, J.-H., *Improved genetic algorithm for the jobshopscheduling problem*, Taiwan, R.O.C., 2005.
[8] Lin, S.-C., Goodman, E.D., and Punch, W.F., *A Genetic Algorithm Approach to Dynamic Job Shop Scheduling Problems*, Technical Report GARAGe97-02-08, Genetic Algorithms Research and Applications Group, Michigan State University, 1997.
[9] Artigues, C., Lopez, P., and Ayache, P.-D., *Schedule generation schemes for the job*-shop problem with sequence-dependent *setup times: dominance properties and computational analysis*, 2003.
[10] Haupt, R., A survey of priority rule-based scheduling, OR Spektrum 11, 3–16, 1989.
[11] Gen, M., and Cheng, R., Genetic algorithms and engineering design. Wiley, New York, 1997.

[12] Hmida, A. B., Huguet, M.-J., Lopez, P., and Haouari, M., *Climbing Depthbounded Discrepancy Search for Solving Flexible Job Shop Scheduling Problems*, 2007.
[13] Ombuki, B. M., and Ventresca, M., *Local Search Genetic Algorithms for the Job Shop Scheduling Problem*, Brock University, Department of Computer Science, Canada, 2002.
[14] Wang, Y. M., Yin, H. L., and Wang, J., *Genetic algorithm with new encoding scheme for job shop scheduling*, Springer-Verlag, London, 2009.
[15] Song, Xiaoyu , "Hybrid particle swarm algorithm for job shop scheduling problems", 2009.
[16] Das S. , Abraham A. , and Konar A., "Particle Swarm Optimization and Differential Evolution Algorithms: Technical Analysis, Applications and Hybridization", Center of Excellence for Quantifiable Quality of Service, Dept. of Electronics and Telecommunication Engineering, Jadavpur University, Kolkata, India, 2007.
[17] Eberhart, R. and Shi, Y.; "Particle Swarm Optimization: Developments", Application and Resources. IEEE, pp: 81-86, 2001.

Author Profile

Wathiq N. Abdullah received the B.Sc. and M.S. degrees in computer science from University of Baghdad in 2001 and 2004, respectively. During 2005-2012, he worked as a lecturer in the Iraqi universities. He received the Ph.D. in Computer Science from the University of Technology in Iraq.