INTERNATIONAL JOURNAL OF MULTIDISCIPLINARY EDUCATIONAL RESEARCH
ISSN : 2277-7881
VOLUME 1, ISSUE 3, AUG 2012

ENHANCED PSO FOR OPTIMIZING MULTI JOB SHOP SCHEDULING

M. Nandhini
Research and Development Centre
Bharathiar University
Coimbatore,TamilNadu -46, India

S. Kanmani
Department of Information Technology
Pondicherry Engineering College
Puducherry-14, India

I. INTRODUCTION

The Multi Job-shop scheduling problem (MJSSP) is one of the combinatorial optimization problems and it is classified as the NP-hard problem. The goal of combinatorial optimization is finding the best possible solution from the set of feasible solutions.

MJSSP is one of the most extremely difficult combinatorial problems ever since the 1950s. The goal of MJSSP is to allocate machines to complete jobs over time, subject to the constraint that each machine can handle at most one job at a time. The complexity of MJSSP increases with its number of constraints and size of search space. The constraints of the problem are so strong, making the valid search space of the problem too complicated.

Historically MJSSP was treated via exact methods or approximation algorithms. Exact methods are based predominantly on the Branch and Bound(BB) method, Dynamic Programming and Constraint Logic programming. Because it is time consuming and only can solve small problems. On the other hand, approximation algorithms ,which are a quite good alternative such as the shifting bottleneck approach (SB), simulated annealing (SA),tabu search (TS),genetic algorithm (GA) and ant colony optimization algorithm (ACO).This new heuristic, called Particle Swarm Optimization (PSO) has been found to be both robust and versatile in handling a wide range of combinatorial optimization problems.

Xia Weijun, Wu Zhiming, Zhang Wei, and Yang Genke (2004) described A New Hybrid Optimization Algorithm for the Job-shop Scheduling Problem [1]. In which, at each generation, jobs are arranged according to the increment order of process on the machine. If one job's process order is the same as the other, the two jobs are arranged randomly. Furthermore, here only after a large number of generations, the solution could be reached.

SONG Cun-li, LIU Xiao-bing, WANG Wei and Bai Xin (2011) presented a Hybrid Particle Swarm Optimization Algorithm for Job-Shop Scheduling Problem. In their work, three neighborhood simulated annealing algorithms are designed and combined with PSO, for each best solution that particle find, simulated annealing is

performed on it to find its best neighbor solution.

Rui Zhang (2011) presented a Particle Swarm Optimization Algorithm based on Local Perturbations for the Job Shop Scheduling Problem[2]. In his work, a local search procedure based on processing time perturbations is designed and embedded into the framework of PSO for MSSP model.

Dr. Liang Gao, Mr. Chuanyong Peng, Mr. Chi Zhou and Prof. Peigen Li(2006) developed a method for Solving Flexible Job-shop Scheduling Problem Using General Particle Swarm Optimization [3]. In their work, a General PSO (GPSO) Algorithm is presented to solve the Flexible Job-shop Scheduling Problem (FJSP). In GPSO, crossover and mutation operations in Genetic Algorithm are respectively utilized by particles to exchange information and search randomly. Tabu Search is used for particles' local search.

Thongchai Pratchayaborirak and Voratas Kachitvichyanukul (2011) published a work on two-stage PSO algorithm for job shop scheduling problem [4]. In their work, they have presented a two-stage particle swarm optimization algorithm (2S-PSO) for job shop scheduling problems to reflect that two stages with multiple swarms are used. The random key representation is used and the schedules are constructed using a permutation with m-repetitions of job numbers.

Metaheuristics is the most popular computational method used for combinatorial problems to find optimal solutions by iteratively trying to improve a candidate solution with regard to a given measure of quality. Particle Swarm Optimization (PSO) algorithm is one of the Meta-heuristics method that is used in this work to find the optimal solutions for the MJSSP instances of larger size.

In this paper, we focus on exploiting particle swarm optimization (PSO) algorithm to achieve the better solution for MJSSP. This new heuristic, called Particle Swarm Optimization (PSO) has been found to be robust in a wide range of combinatorial optimization problems. The optimization mechanism of the traditional PSO is analyzed and a general optimization model based on swarm intelligence is proposed. Experimental results of benchmark problems of Lawrence (1984) from OR library and Fisher and Thompson(1963) (Beasley J E (1990)) show the effectiveness of PSO on MJSSP.

This paper is organized as follows. In Section.2, problem description of job-shop scheduling and mathematical representation of its constraints is mentioned. The general description of Particle Swarm Optimization is discussed in Section 3.The proposed method and its implementation are presented in Section 4. Empirically evaluated experimental results on a set of typical instances are shown in Section 5. Conclusions and final remarks are discussed in Section 6.

II. MULTI-JOB SHOP SCHEDULING PROBLEM

Multi Job-shop scheduling problem (MJSSP) is considered as hard combinational optimization problem and it has been the subject of a significant amount of literature in the Operations Research (OR) areas. The MJSSP consists of *n* jobs and *m* machines. Each job must go through *m* machines to complete its work and it is considered that one job consists of *m* operations. Each operation uses one of *m* machines to complete one job's work for a fixed time interval. Once one operation is processed on a given machine, it cannot be interrupted before it finishes the job's work. The sequence of operations of one job should be predefined and may be different for any job. In general, one job being processed on one machine is considered as one operation noted as O_{ij} (means j^{th} job being processed on i^{th} machine, $1 = j = n$,$1 = i = m$)*(Garey et al. 1976 and Lawler et al. 1993)*. Each machine can process only one operation during the time interval.

The objective of MJSSP is to find an appropriate operation permutation for all jobs that can minimize the makespan Cmax i.e.,

- To Minimize Final completion time
- To Minimize Idle time of machines
- To Minimize Waiting time of job.

The maximum completion time of the final operation in the schedule of n×m operations with minimum waiting time of jobs and machines is called makespan. For an *n*×*m* JSSP, the problem can be modeled by a set of *m* machines, denoted by $M=\{1,2,\ldots,m\}$, to process a set of *n*×*m* operations, denoted by $O = \{1,2,\ldots,(n\times m)\}$ (Lin *et al*. 2009).

Where,

n : number of jobs

m : number of operations for one job

Oi : completed time of operation *i* (i=1,2,……($n\times m$))

ti : processing time of operation *i* on a given machine

Cmax : makespan

The problem can be understood with its known constraints (mandatory &optional) / assumptions as listed below.

- No machine may process more than one job at a time.
- No job may be processed by more than one machine at a time.

· The order in which a job visits different machines is predetermined by technological constraints.

· Different jobs can run on different machines simultaneously.

· At the moment *T*, any two operations of the same job cannot be processed at the same time.

· Processing time on each machine is known.

· Idle time of machines may be reduced

· Waiting time of jobs may be reduced.

Thus, the MJSSP is to allocate machines to complete jobs over time, subject to the above constraint. The complexity of MJSSP increases with its number of constraints and size of search space. The constraints of the problem are so strong, making the valid search space of the problem too complicated.

III. PARTICLE SWARM OPTIMIZATION ALGORITHM

Particle swarm optimization (PSO) is a metaheuristics, population based stochastic optimization technique for continuous non-linear problems developed by Dr. Eberhart and Dr. Kennedy in 1995, inspired by social behavior of bird flocking or fish schooling for foods. The concept of particle swarms, although initially introduced for simulating human social behaviors, has become very popular these days as an efficient search and optimization technique. The particle swarm optimization (PSO), as it is called now, does not require any gradient information of the function to be optimized, but uses only primitive mathematical operators and is conceptually very simple.

In PSO, there is a population called a swarm, and every individual in the swarm is called a particle which represents a solution to the problem. All of particles have fitness values which are evaluated by the fitness function to be optimized, and have velocities which direct the flying of the particles. The particle flies in the D-dimensional problem space with a velocity which is dynamically adjusted according to the flying experiences of its own and its colleagues. The basic Principle of Particle swarm move towards the best position in search space, remembering each particle's best known position (pbest) and global (swarm's) best known position (gbest).

Algorithm begins with a set of solutions called swarm of particles. For each particle in the swarm, velocity and position is calculated with their constraints to obtain feasible solutions. Several such particles are formed in each generation. The PSO operators are used to preserve the positions. A fitness function is evaluated for each schedule generated. The new generation of solutions thus formed is most likely to be

better than the previous ones. This process of forming new generation of solutions is continued until a best fitness value is obtained.

In the evolutionary process, the system is initialized with a population of random solutions and searches for optima by updating generations. In every iteration, each particle is updated by following two "best" values. The first one is the best solution (fitness) it has achieved so far. This value is called pbest. Another "best" value that is tracked by the particle swarm optimizer is the best value, obtained so far by any particle in the population. This best value is a global best and called gbest.

After finding the two best values, the particle updates its velocity and positions with following equation (a) and (b).

$$v[] = v[] + c1 * rand() * (pbest[] - present[]) + c2 * rand() * (gbest[] - present[]) \quad (a)$$

$$present[] = persent[] + v[] \quad (b)$$

where *v[]* is called the velocity, of the particle *i*, represents the distance to be travelled by the particle from its current position. *persent[]* represents the current particle position. *pbest[]*is called *pbest* (local best solution), represents i^{th} particle's best previous position and *gbest[]*which is called *gbest* (global best solution), represents the best position among all particles in the swarm.*rand ()* is a random functions between [0,1]. *c1, c2* are the acceleration constants which represent the weight of the stochastic acceleration terms that pull each particle toward *pbest* and *gbes*t positions

For equation (a), the first part represents the inertia of previous velocity. The second part is the "cognition" part, which represents the private thinking by itself. The third part is the "social" part, which represents the cooperation among the particles. Fig. 1 illustrates the PSO algorithm:

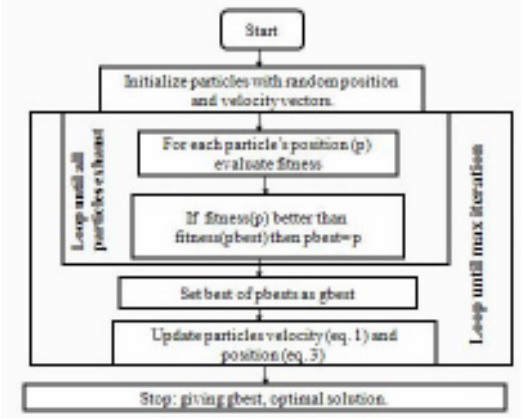

Fig. 1 Design of PSO Algorithm

I. PROPOSED SYSTEM

Initially, we start with a possible solution (particle) for the schedules randomly filled. This process is repeated to get 50 particles. This set of particles is our initial swarm. The particles evolve through successive iterations, called generations. During each generation, the particles are evaluated, using some measures of fitness. To create the next generation, new particles are formed by the following steps (1) evaluate (pbest) best particle's known position (2) evaluate global (swarm's) best known position (gbest) (3) Calculate velocity and position for each particle. (4) Evaluate the position for the same number of job occurrences and find the proper schedule with their constraints (5) Calculate the fitness for the particle obtained. Thus, the new generation is formed and it is repeated. After several generations, the algorithm converges to best particle, which hopefully represents the optimum solution to the problem.

The representation, fitness function and other PSO operators that are used in our implementation are discussed in this section. Usually initialization is assumed to be random. The new generation typically involves the above steps to yield optimum solution.

A. State Representation

State formation takes number of jobs, number of operations for each job, sequence of operation of each jobs, processing time of each job's operation and allotment of operation over machines as inputs. In our work, each state is represented as an array of structures. Each structure consists of job name and its operation as members.

Job No	Oper No	Job No	Oper No	.....	Job No	Oper No

Fig. 2 General Representation of MJSSP

For 3 Jobs and 3 Operations, the schedule is represented as follows in Fig.3 which the Operation sequences should not change:

O_{11}	O_{12}	O_{13}	O_{21}	O_{22}	O_{23}	O_{31}	O_{32}	O_{33}

Fig. 3 Example of Schedule

O_{11} -> Means first job's first operation

O_{23} -> Means second job's third operation

Since, 3 jobs x 3operations, the array structure should be of 9 positions with all jobs operations.

A. Feasible Solution Generation

A Feasible schedule is generated by the constructive heuristics process. First operation in a schedule can be scheduled if it is been first operation of any of the jobs. Following this, unscheduled operations of all the jobs are scheduled by verifying operation consistencies and capacity constraints. In the following example, the heuristic process is clearly mentioned. The Operation sequence of any Job should not change in the schedule. It should start from 1 and ends with m (no. of operations)

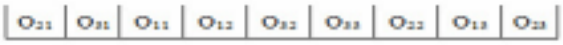

Fig. 4 Feasible Schedule

O_{11} -> Means first job's first operation

O_{23} -> Means second job's third operation

Since, 3 jobs x 3operations, the array structure should be of 9 positions with all jobs operations.

For n jobs on m machines MJSSP, the schedule is created in n x m dimensions.

A. Fitness Function Evaluation

The evaluation of individual in the swarm is commonly regarded as the most computational demanding step. This step is executed in every generation for all the individuals. The evaluation of an individual can be done in isolation from the rest of the population, and no communication with the other individuals of the population is required or desired. The identification of good or bad solutions in a population is usually accomplished according to a solution's fitness as in eqn. c. The essential idea is that a solution having a better fitness must have a higher probability of selection.

Min Z = Min (Makespan+

Jobs Waiting Time+ ———————— *eq.* *(c)*

Machines Waiting Time)

Where,

Job No. i : 1..n

Operation No. j : 1..m; $j_1 < j_2 < j_3 < \ldots < j.m$

In this work, the fitness function is a kind of objective function that quantifies the optimality of a solution, so that the particle may be ranked against other particle in the population. In our case, the fitness for the MJSSP is evaluated using Gantt chart. From the Gantt chart, the processing time of all jobs, idle time of a machine and idle of

Job is found. The sum of all these three is said to be fitness for a particle.

B. Swarm Initialization

The swarm is initialized with a search space of *n* x *m* dimensions for a problem of *n* jobs on *m* machines. Generally particles' positions and velocities in initial swarm are generated randomly. The schedule (job, operation) is converted into their corresponding position number for evolutionary process as follows:

The method for finding position is introduced with the following steps and shown in Fig.5..

If Job=1, Oper=1 *Position=1*

If Job=1, Oper>1 *Position=Operation*

If Job>1, Oper=1 *Position=(Job*No. of Oper) – (No. of Oper – 1)*

If Job>1, Oper>1 *Position=(Job*No. of Oper) – (No. of Oper – 1)+(Oper-1)*

For Example,

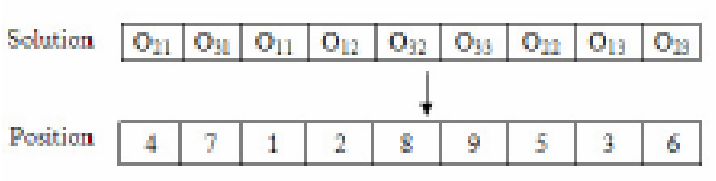

Fig. 5 Position Evaluating

The velocity for all the particles is initialized between $[-v_{max}, v_{max}]$ and pbest and gbest is initialized with the first particle fitness value.

A. pbest Evaluation

The particle's best known position achieved so far in their respective iteration is said to be pbest. It is obtained by comparing the previous pbest value with the fitness of the current particle. If it is better the pbest is changed with the new better fitness.

B. gbest Evaluation

The global best known position in swarm achieved so far in each iteration is said to be gbest. It is obtained by comparing the previous gbest value with the minimum fitness obtained in current iteration. If it is better the gbest is changed with the new

better fitness.

C. PSO Operators

Once the schedule representation and the fitness function are defined, PSO proceeds to initialize a population of solutions randomly and then improve through repetitive application of pbest and gbest.

1) Velocity Updation : In each iteration, after finding the two best values, the particle updates the velocity with the equation (a).

The equation (a) helps the particle move in same direction to achieve optimum. The velocity equation has three parts. The first part represents the inertia of previous velocity, forces the particle to move in same direction. The second part is the "cognition" part, which represents the private thinking by itself and forces the particle to go back to the previous best position. The third part is the "social" part, which represents the cooperation among the particles, forces the particle to move to the best previous position of its neighbors. The maximum velocity v_{max} is said to number of jobs. If the velocity exceeds $v_{max,}$ then it is assigned to $v_{max.}$

2) Position Updation : The position vector represents jobs' arrangement on all machines. The maximum position p_{max} is said to nxm (no. of jobs * no. of operations). If the position exceeds $p_{max,}$ then it is assigned to $p_{max.}$ The results of a particle's position may have a meaningless job number as a real value such as 3.125. So, the real optimum values are round off to its nearest integer number. The computation results of equation (b) will generate repetitive code (job number), i.e. one job is processed on the same machine repeatedly. It violates the constraint conditions in MJSSP and produce banned solutions. Banned solutions can be converted to legal solutions by modification. The process of modifying solutions is as follows:

- Check a particle and record repetitive job numbers on every machine.
- Check absent job numbers on every machine of a particle.
- Sort absent job numbers on every machine (of a particle) according to increment order of their processes.
- Substitute absent job numbers for repetitive codes on every machine of a particle from low dimension to high dimension accordingly.

For example, the following is the positions obtained after applying to the formula.

Position	1	4	4	5	5	6	8	6	9

This position should be scheduled using constructive heuristics process. In the above figure, the position 4 corresponds to job 2 first operation and the same position 4 is repeated, so their next operation is scheduled. Position 5 is repeated twice and job 2 operations are completed. So, the Modifying solution procedure is applied and unscheduled least job operation is scheduled. Likewise, the schedule is obtained for the above positions.

These steps are repeated for further generations, to find the optimum value for MJSSP.

PSO Algorithm

1. *[Start] Generate random population of n particles (suitable solutions for the problem).*
2. *[Fitness] Evaluate the fitness f(x) of each particle x in the swarm.*
3. *[New population] Create a new swarm by repeating the following steps until the new swarm is complete.*

· *[pbest] If the fitness value is better than the best personal fitness value in history, set current value as a new best personal fitness value.*

· *[gbest] Choose the particle with the best fitness value of all the particles, and if that fitness value is better than current global best, set as a global best fitness value*

· *[velocity] Calculate particle velocity according velocity change equation*

· *[position] Update particle position according position change equation*

4. *[Replace] Use the new generated swarm for a further run of the algorithm.*
5. *[Termination] If the end condition is satisfied, stop, and return the best solution in the current swarm.*
6. *[Loop] Go to step 2.*

I. COMPUTATIONAL RESULTS

The proposed PSO implemented on MJSSP is tested on two classes of standard test problems: Fischer and Thompson (1963) instances FT06 and FT10 and Lawrence(1984) from OR library (LA1 to LA10, LA16 to LA20) (BeasleyJ.E 1990)and the optimal values obtained are tabulated in Table 1 and are compared against the best known values found for the respective instances.

The algorithm is implemented in Java Programming Language and run it on the Intel Pentium Core 2 Duo 3GHz Processor and 2 GB RAM configuration system with Windows XP as the platform.

TABLE I

Computational Results

Sl.No.	Instance Name	Instance Size	Best Known Value	Proposed PSO
1	FT01	6 x 6	55	55
2	FT10	10 x 10	930	935
3	LA01	10 x5	666	666
4	LA02	10 x 5	655	659
5	LA03	10 x 5	597	597
6	LA04	10 x 5	590	590
7	LA05	10 x 5	593	593
8	LA06	15 x 5	926	926
9	LA07	15 x 5	890	890
10	LA08	15 x 5	863	863
11	LA09	15 x 5	951	951
12	LA10	15 x 5	958	958
13	LA16	10 x 10	945	945
14	LA17	10 x 10	784	787
15	LA18	10 x 10	848	848
16	LA19	10 x 10	842	842
17	LA20	10 x 10	902	902

From Table.1 , the performance of the proposed PSO is shown with optimal values same as best known values for most of the instances taken for testing except FT10,LA02 and LA17. Hence , it is proved that the proposed PSO could give robust results for MJSSP problem of any size.

I. CONCLUSION

Thus, we have hereby come up with multi objective optimal solution using the PSO algorithm There is a guaranteed improvement in the solutions due to the process of repairing the solutions. Also, repairing the solution for its infeasibility evidently helps in reduce the response time of this process. Hence, a user friendly and easily understandable solution for multi constrained combinatorial problem with multiple objectives is possible. As future enhancement of this project, this can be hybridized with local search .

REFERENCES

[1] Xia Weijun, Wu Zhiming, Zhang Wei, Yang Genke, "A New Hybrid Optimization Algorithm for the Job-shop Scheduling Probem", In Proc. of the American Control Conference Boston, pp. 5552-5557, June 30 - July 2, 2004.

[2] Rui Zhang, "A Particle Swarm Optimization Algorithm based on Local Perturbations for the Job Shop Scheduling Problem", International Journal of Advancements in Computing Technology, Vol. 3, No. 4, pp. 256-264, May 2011.

[3] Dr. Liang Gao, Mr. Chuanyong Peng, Mr. Chi Zho, Prof. Peigen Li, "Solving Flexible Job-shop Scheduling Problem Using General Particle Swarm Optimization", The 36th CIE Conference on Computers & Industrial Engineering, pp.3018-3027,2006.

[4] Thongchai Pratchayaborirak, Voratas Kachitvichyanukul," A two-stage PSO algorithm for job shop scheduling problem", International Journal of Management Science and Engineering Management, pp.84-93, 2011.

[5] M. Nandhini, S. Kanmani, Rajesh Kumar Sahoo, "Heuristics Supported Local Search for Optimization of Multi Job Shop Scheduling" , IJCA Special Issue on Artificial Intelligence Techniques - Novel Approaches & Practical Applications, pp. 34-39, 2011.

[6] Sohrab Khanmohammadi, Hamed Kharrati, "A New Hybrid evolutionary Algorithm for Job-shop Scheduling Problems", ISBN: 978-960-474-260-8, Applied Mathematics and Informatics, pp.51-56, 2010.

[7] Wen-liang Zhong, Jun Zhang, Wei-neng Chen, "A Novel Discrete Particle Swarm Optimization to solve Travelling Salesman Problem",IEEE Congress on Evolutionary Computation, pp.3283-3287, 2007.

[8] Zhigang Lian, Bin Jiao, Xingsheng Gu, "A similar Particle Swarm Optimization algorithm for job-shop scheduling to minimize makespan", Applied Mathematics and Computation 183, pp.1008–1017, 2006.

[9] Kun Tu, Zhifeng Hao, Ming Chen, "PSO with Improved Strategy and Topology for Job Shop Scheduling", ICNC, Part II, LNCS 4222, pp. 146 – 155, 2006.

[10] Elnaz Baghal Azardoost, Nrges Imanipour, "A Hybrid Algorithm for Multi Objective Flexible Job Shop Scheduling Problem", In Proc of the International Conference on Industrial Engineering and Operations Management, pp. 795-801, January 22 – 24, 2011

[11] http://www.swarmintelligence.org

[12] Beasley J E ," OR-library: Distributing test problems by electronic mail", Journal of the Operational Research Society,41(11),1069–1072, 1990.

[13] Fisher H. and Thompson G.L.," Probabilistic Learning Combinations of Local Job-Shop Scheduling Rules in: Industrial Scheduling" , Prentice-Hall, Englewood Cliffs, NJ, pp. 225-251, 1963.