Research Journal of Applied Sciences, Engineering and Technology 13(5): 409-415, 2016
DOI:10.19026/rjaset.13.2959
ISSN: 2040-7459; e-ISSN: 2040-7467
© 2016 Maxwell Scientific Publication Corp.
Submitted: April 2, 2016 Accepted: May 23, 2016 Published: September 05, 2016

Research Article

A Comparison of Genetic Algorithm and Particle Swarm Optimisation to Minimize the Makespan for Parallel Line Job Shop Scheduling

P. Ravichandran, K. Krishnamurthy and B. Meenakshipriya
Department of Mechatronics Engineering, Kongu Engineering College, Erode, Tamilnadu, India

Abstract: This paper describes the implementation of Genetic Algorithm (GA) and Particle Swarm Optimization (PSO) algorithm to minimize the total completion time of jobs called makespan in parallel line Job Shop Scheduling Problem (PJSSP). PJSSP is one of the scheduling methods in manufacturing industry for increasing plant utilization, reducing cycle time and to find the optimal allocation of jobs in multiple processing lines. Each job is assigned to a particular line and the job should be completed only in that assigned line. Also, the job should be processed in a particular order. PJSSP is always a challenging task in the combinatorial research and it requires a heuristic approach. Results show that the performance of PSO is superior to GA in order to find optimal solution with minimum makespan for PJSSP.

Keywords: Genetic Algorithm (GA), makespan, Parallel line Job Shop Scheduling Problem (PJSSP), Particle Swarm Optimisation (PSO), Processing time, Setup time

INTRODUCTION

In the manufacturing system, scheduling and planning a production order plays an important role. The scheduling and planning problems are usually combinatorial. Scheduling problems are defined as the process of assigning a set of jobs to resources over a period of time (Zhang *et al*., 2013). In recent years, scheduling problems exist in real world industrial situations. Performance criteria such as machine utilization, manufacturing lead times, processing time, inventory cost and meeting due date, customer satisfaction and quality of products are dependent on the efficient way the jobs are scheduled (Tang and Dai, 2015). Hence, it becomes increasingly important to develop an effective job shop scheduling approach.

A job shop environment consists of 'n' jobs and each job has a given machine route, where some machines are missed and repeated (Geyik and Cedimoglu, 2004). A parallel line scheduling problem has more than one processing line for jobs to be processed. The processing time of a job is different for each machine. Each job is assigned to only one particular line and is processed to complete in that line. Further, each job enters their respective lines at the same time and is processed in parallel manner. Thus, the parallel line job shop scheduling is challenging and time consuming (Haq *et al*., 2008). Job Shop Scheduling (JSS) problem involves a mission of a set of tasks to the workstations or machines in a particular sequence. Hence, it requires efficient evolutionary techniques to overcome the above issues.

During the past years of computational intelligence, different evolutionary algorithms are proposed in solving job shop scheduling problems (Gromicho *et al*., 2012). These evolutionary algorithms include Tabu Search method (Solimanpur and Elmi, 2013), Ant Colony Optimisation (Cheng *et al*., 2013), Genetic algorithm (Musharavati and Hamouda, 2011), Particle Swarm Optimisation (Behnamian, 2014) and Simulated Annealing (Elmi *et al*., 2011). Kennedy and Eberhart have proposed better result using Particle Swarm Optimization (PSO) than different optimization algorithms. PSO follows a collaborative population-based search, which models over the social behavior of bird flocking and fish schooling. PSO has many advantages over other heuristic techniques such as few lines of computer code, primitive mathematical operators, escaping local optima and easily modified. From earlier studies, it is understood that PSO has not been implemented for parallel line job shop scheduling problem. Hence, this work is focused towards the development of PSO algorithm for parallel line job shop scheduling by considering the setup time, process time and the quantity of jobs.

Then, the same parallel line scheduling problem is solved by the implementation of Genetic algorithm by considering same set of data. Thus, the obtained results

Corresponding Author: P. Ravichandran, Department of Mechatronics Engineering, Kongu Engineering College, Erode, Tamilnadu, India, Tel.: +919965998989

Table 1: Setup time matrix (S)

Jobs	1	2	3	4	5	6	7	8	9	10
1	-	6	5	9	11	9	3	10	6	4
2	6	-	2	7	4	6	4	2	2	10
3	4	7	-	4	7	4	9	1	11	9
4	9	11	9	-	12	7	2	6	7	4
5	11	9	7	7	-	11	14	13	3	2
6	3	4	11	14	9	-	7	12	6	8
7	1	12	6	7	7	11	-	11	8	13
8	7	13	14	4	13	9	4	-	14	15
9	8	9	11	8	6	8	8	10	-	8
10	15	5	9	5	6	5	10	6	8	-

are compared with the result of PSO algorithm. The parallel line scheduling problem was taken as the empirical data by Haq *et al.* (2008). They have implemented the genetic algorithm to solve the parallel line job shop scheduling without considering the quantity of jobs. Hence, in this proposed work, GA and PSO algorithms are designed and implemented for parallel line job shop scheduling by considering the quantity of jobs.

MATERIALS AND METHODS

When and where this study was conduct*: Kongu Engineering College, Erode, India.

Parallel line job shop scheduling problem: There are totally 'n' number of jobs and 'm' number of machines with 'm' number of operations in a specific order. Each operation is allocated to a particular machine with a fixed processing time and fixed setup time. In the parallel line job shop scheduling problem, more than one processing lines are present. Every processing line has non-identical machines in a specified order. Similar set of machines with same order is maintained for all the processing lines. The following are the constraints for scheduling problem (French, 1982):

- Each job must be processed through each machine once and only once.
- Each job should be processed through the machine in a specified order.
- Each operation must be executed uninterrupted on a given machine.
- Each machine can only handle at most one operation at a time.

In job shop scheduling problem, the sequence always follows the most priority rule and technique (Panneerselvam, 2012). Some of the priority rules are shortest processing time, first come first serve, most-work remaining, most operations remaining and least work remaining and random. The setup time matrix 'S' is same for all the lines. After a job is processed, setup time (S) is necessary when a new job enters in the processing line and is shown in Table 1. The Process time matrix 'P' shows the processing time for the different jobs in each processing lines as shown in

Table 2: Processing time matrix (P)

jobs	Lines	M1	M2	M3
1	1	12	14	16
	2	23	22	10
	3	20	19	11
2	1	16	17	9
	2	13	15	16
	3	15	22	22
3	1	10	25	25
	2	11	21	13
	3	24	21	12
4	1	21	18	17
	2	19	14	8
	3	14	9	17
5	1	19	12	10
	2	11	16	19
	3	11	21	21
6	1	16	22	24
	2	13	17	22
	3	9	23	20
7	1	16	21	10
	2	13	13	18
	3	21	14	12
8	1	22	19	17
	2	7	7	21
	3	12	11	18
9	1	16	21	20
	2	10	23	22
	3	15	21	21
10	1	9	25	20
	2	11	17	21
	3	11	18	16

Table 2. The quantity (Q) of each jobs are shown in Table 3 (Haq *et al.*, 2008).

Fitness function: The entire performance of proposed algorithm such as efficiency, convergence speed and optimisation accuracy depends on the fitness function which supervises the optimum search of the solution within the search space.

The fitness function chosen is either to minimize the preference constrains or to maximize the domain constrains. The objective of the parallel line job shop scheduling problem is to minimize the Makespan (Cmax). The variables are listed as follows:

n	: Number of jobs
l	: Number of processing lines
m	: Number of machines on each processing line
Q	: Quantity of each job to be processed

Table 3: Quantity of each job (Q)

Jobs	J1	J2	J3	J4	J5	J6	J7	J8	J9	J10
Quantity	54	32	78	61	65	47	29	55	60	72

$P_{i,j,k}$: Processing time of each job in with respect to all machines in each line
$S_{i,j}$: Setup time of each job with respect to all other jobs
$T_{i,j,k}$: Starting time for each job in each lines

There are 'n' jobs J = {J1, J2, J3,……, Jn} to be processed through 'm' machines M = {M1, M2, M3, …, Mm} each job Ji has 'm' operations Oi = {Oi, 1, Oi,2, Oi,3, …, Oi, m} where i denotes the job number. According to the taken priority rule technique first come first served basis, Oi, j must be processed after Oi,j-1 where j is the job number. The process time matrix, Pi, j, k, is the processing time for i^{th} job in j^{th} machine in processing line 'k'. Ti, j, k is the starting time for ith job in jth machine in processing line 'l' in line 'k'. Quantity of each job is Qi = {Q1, Q2, Q3……Qq} and Si, j setup time for the jobs which is equal for all the processing lines. The fitness function is calculated using the Eq. (1):

$$C_{max} = \min (\max \{C_{i,j,k}\}) \quad (1)$$

where,

$$C_{i,j,k} = \sum_{Q=1}^{q} (T_{i,j,k} + S_{i,j} + P_{i,j,k})$$

$$i = 1, 2,\dots n;\ j = 1, 2,\dots m;$$
$$k = 1, 2,\dots l,\ Q = 1, 2, \dots q$$

$$T_{i,j,k} = \begin{cases} 0 \text{ if } j = 1 \\ C_{i,j-1,k} \text{ else where} \end{cases}$$

Makespan calculation: Makespan is calculated as the follows in Eq. (2):

Makespan=Total Processing Time
+Total Setup Time (2)

Processing time is calculated for each job separately. For example, the processing time of Job1 in Line 1 is calculated using the Eq. (3).

$(\text{Processing Time})_{J1}$ = Total quantity of Job1 * Processing time of Job1
in all machines of Line1 (3)

Similarly the processing times for all jobs in all lines are calculated. Now, the total processing time of all jobs in a line is calculated by adding the individual processing times of all jobs in that line as shown in Eq. (4). For example, if J1, J2 and J3 are in line1, then:

The total processing time of all jobs in line1 = $(\text{Processing Time})_{J1}$ $+(\text{Processing Time})_{J2}+(\text{Processing Time})_{J3}$ (4)

Finally, the respective set up times of the jobs is added to the processing time of that line and it is called as makespan of that line. Similarly makespan of all lines are calculated. In which, the largest makespan is taken as the makespan of the entire process.

METHODS

Genetic algorithm: Genetic Algorithm (GA) is one of the non-conventional optimisation techniques to solve combinatorial problem. It focuses on the principle of survival of the fittest. It resembles the behaviour of natural evaluation and genetics. Genetic algorithm is applicable to real world problems, if they are suitably encoded. Generally, fitness function is used to evaluate the genome. Genome has 'N' number of individual chromosome known as population size. After this, the fitness evaluation chromosomes pass through the reproduction phase. Then the crossover and mutation are applied to produce new population. In the reproduction phase, elimination of weak chromosomes is done and fitted chromosomes are retained for the next generation.

Chromosome representation in GA: Chromosome representation can be used for all the individual genomes in the genetic algorithm. Each individual chromosome gives a complete solution to the problem for the optimisation to carry out. Initial population is generated randomly using permutation of job. Figure 1 shows an example of chromosome in a Genome.

The following parameters are used for parallel line job shop scheduling problem:

(i) Number of jobs (n) = 10
(ii) Number of processing lines (l) = 3
(iii) Number of machines in each line (m) = 3
(iv) Population size = 100

Selection: The selection operator is responsible for a chromosome to migrate from the current generation of population to the next generation population growth. Before making it into the next generation's population, selected chromosomes undergo crossover and/or mutation phase. The new offspring chromosome (s) are produced to the next generation's population. In earlier studies Roulette wheel is widely used selection operator.

Crossover: Crossover is used to produce new offspring. Crossover probability determines the number of cross over operation. There are many different kinds

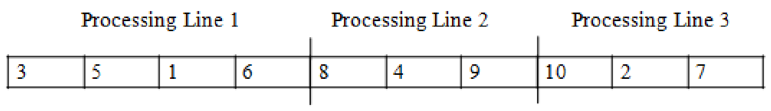

Fig. 1: Chromosome representation of sequence

of crossover and the most common type is single point crossover. In single point crossover, first choose a locus, then decide from that point, where swapping operation going to perform from one parent to the other. It was noted that, children take one section of the chromosome from each parent. The point at which the chromosome is broken depends on the randomly selected crossover point. From the earlier analysis, crossover probability gives good solution in the range of 0.7 to 0.9.

Pseudo-code of GA:

```
Input: PLJSSP data sheet, GA parameters
Output: A near optimal schedule, minimum makes pan
Begin
    Initialize
      a) Initialize the GA parameters:
         Population size (n), iteration number (Iter),
         Cross over probability (Pc), mutation
         probability (Pm);
      b) Initialize population of 'n' chromosomes by
         randomly sequencing genes;
      Compute the fitness function for all 'n'
         chromosomes;
Repeat (outer loop: evolve population)
    Repeat (inner loop: generate new individuals)
        Select two parent chromosomes
        Apply crossover
        Apply mutation
Until 'n' new individuals are generated
      Select 'n' elite members for the next generation
      Evaluate fitness function;
Until termination conditions are reached;
      Output a near optimal schedule and makespan;
End
```

Mutation: Mutation operation used to produce randomness in the search space. Mutation is performed with in particular chromosome. Mutation probability determines number of mutation operation in reproduction. Typically the range of mutation probability is 0.001 to 0.1. For this problem, the mutation probability is taken as 0.01.

Particle Swarm Optimisation Algorithm (PSO): PSO is used for continuous solving of combinatorial optimisation problem due to the discrete nature of job shop scheduling problem. Smallest position value rule is applied to convert continuous PSO into Discrete PSO, borrowed from random key genetic algorithm. Each particle moves around a 'D' dimensional space with the velocity. Each iteration this velocity is updated and the corresponding position also calculated with the help of velocity. In this process, initial swarm position is determined randomly.

The continuous position values are generated and updated by the following Eq. (5):

$$\text{Position}_{i,j} = \text{Position}_{min} + (\text{Position}_{max} - \text{Position}_{min}) * R_1 \quad (5)$$

where,
$\text{Position}_{min} = 0.0$
$\text{Position}_{max} = 4.0$
R_1 = A uniform random number in between 0 to 1

Initial velocity of the particles also generated and updated by the equation 6,

$$\text{Velocity}_{i,j} = \text{Velocity}_{min} + (\text{Velocity}_{max} - \text{Velocity}_{min}) * R_2 \quad (6)$$

where,
$\text{Velocity}_{min} = -4.0$
$\text{Velocity}_{max} = +4.0$
R_2 is a uniform random number in between 0 to 1

The velocity range is one of the constraints in PSO which ranges from -4 to +4. After the initial velocity and position, it is updated for each iteration as given by the Eq. (7) and (8):

$$\text{Velocity}_{i,j} = W * \text{Velocity}_{i,j} + C_1 * R_1 * (\text{Pbest}_{i,j} - \text{Position}_{i,j}) + C_2 * R_2 * (\text{Gbest}_{i,j} - \text{Position}_{i,j}) \quad (7)$$

$$\text{Position}_{i,j} = \text{Velocity}_{i,j} + \text{Position}_{i,j} \quad (8)$$

where,

$$W = W\text{max} - \text{iteration} * (W\text{max} - W\text{min}) / \text{Max iteration}$$

where 'W' = inertia weight ranges from Wmax to Wmin (1 to 0.2). Initially inertia weight is high, but it gradually decreased from Wmax to Wmin when iterations go on. In the velocity update equation, 'W' is the only variable to control. C_1 is a self-learning factor equal to 2, which is how much a particle can believe its own best fitness Pbest. C_2 is a social-learning factor equal to 2, which is how much a particle can trust Gbest. R_1 and R_2 is a uniform random numbers in between 0 to 1. Each particle records its own best position as Personal best (Pbest). Best of Pbest is taken as Global best (Gbest).

Pseudo-code of PSO:

```
Initialize parameters
Initialize velocity
Initialize position
Find permutation of jobs
Evaluate fitness of all particles
Do
{
    Find position best
    Find global best
    Update velocity
    Update position
    Find permutation of jobs
    Evaluate fitness of all particles
    Local search to find optimal solution
    } While (until termination)
```

Smallest position value Rule: Smallest Position Value rule (SPV) is used to obtain the sequence of jobs from corresponding position values (Bean, 1994). According to SPV rule, the position value is sorted in ascending order. Then the corresponding position value index is fixed from the representation job sequence. The above sequence is shown in Table 4.

The example of sequence of 'n' jobs as per SPV rule is shown in Fig. 2. First four jobs go to processing line1, next three jobs goes to processing line2, remaining 3 jobs goes to processing line3. Each processing line gives makespan for corresponding job sequence. Highest value from the three processing line is the makespan of the particle. By this way, makespan is calculated for all the particles in the swarm.

Local search for PSO: Local search is used to intensify the search towards optimal or near optimal solution. In the proposed work, Variable Neighborhood Search (VNS) is used (Tasgetiren *et al*., 2007). Two operations involved in VNS are insert and interchange.

Table 4: SPV Rules to represent job

Position index	1	2	3	4	5	6
Position value	3.5	2.8	0.74	3.9	1.62	0.59
Job sequence	6	3	5	2	1	4

Table 5: Insert operation in VNS

Position index	1	2	3	4	5	6
Sequence before insert	6	3	5	2	1	4
After insert	6	5	2	1	3	4

Table 6: Interchange operation in VNS

Position index	1	2	3	4	5	6
Sequence before interchange	6	5	2	1	3	4
After interchange	6	3	2	1	5	4

Table 7: Comparative makespan results of GA and PSO algorithms

				GA		PSO	
Iteration No.	No. of Jobs	No. of achines	No. of Lines	Makespan	Sequence obtained	Makespan	Sequence obtained
1	10	3	3	3955	6-7-5-1	3911	9-5-2-1
					8-10-4		8-4-10
					9-2-3		6-3-7
25	10	3	3	3874	5-2-7-10	3662	7-1-4-5
					1-6-9		2-6-3
					4-3-8		9-8-10
50	10	3	3	3766	1-5-7-8	3463	1-6-5-2
					10-3-2		7-4-3
					4-9-6		10-8-9
75	10	3	3	3525	6-5-7-1	3459	5-2-1-6
					2-3-4		4-3-7
					9-8-10		8-9-10
100	10	3	3	3473	2-1-6-5	3459	5-2-1-6
					7-3-4		4-3-7
					8-9-10		8-9-10
125	10	3	3	3473	2-1-6-5	3459	5-2-1-6
					7-3-4		4-3-7
					8-9-10		8-9-10

Table 8: Best sequence and makespan of GA and PSO

Optimisation algorithm	Best sequence		Best Makespan
GA	Processing line 1	2-1-6-5	3473
	Processing line 2	7-3-4	
	Processing line 3	8-9-10	
PSO	Processing line 1	5-2-1-6	3459
	Processing line 2	4-3-7	
	Processing line 3	8-9-10	

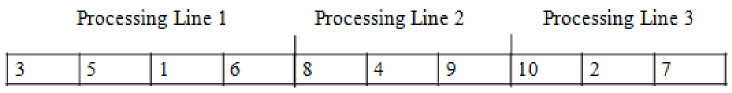

Fig. 2: Assignment of jobs in processing line

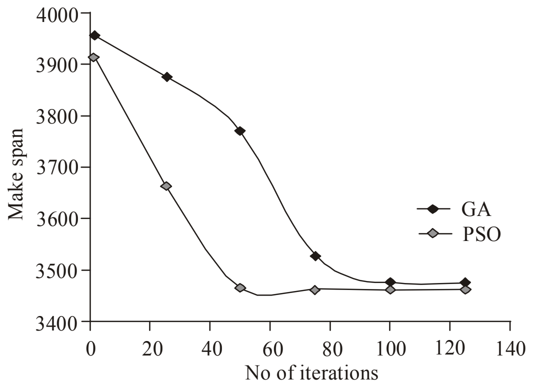

Fig. 3: Performance of GA and PSO algorithms

Local search is not used in continuous position values, for permutation of jobs kind of problem which is very useful. In insert operation, two random numbers (N1, N2) are generated and are not equal. The position of first generated number is removed and reinserted in the second generated random number (e.g., N1 = 2, N2 = 5) (Table 5).

In interchange operation, generated random numbers, N1 and N2, which are not equal and are swapped (Table 6).

RESULTS AND DISCUSSION

The comparative Makespan results of GA and PSO for parallel line job shop scheduling problem is given in Table 7 and the performance for each iteration are shown in Fig. 3.

Basically GA is a local search algorithm and PSO is a local search algorithm. To improve the local search ability of PSO algorithm, VNS technique is implemented with basic PSO algorithm. Due to which, for every iteration of the algorithm, PSO performed with both global and local search ability. In Fig. 3 it is shown that when the iteration increases, convergence leads to the best solution. At the same time, by considering the quantity of jobs, the results shown the best sequence for parallel line job shop problem.

CONCLUSION

In the proposed work, an approach based on comparison of Genetic Algorithm and Particle Swarm Optimisation for solving parallel line job scheduling problem was presented. The results shows that when quantity of jobs considered in an account then the optimisation algorithms gives best solutions for parallel line job shop scheduling for each of the given processing lines. It is applied and executed to any number of lines, jobs and machines. The best solution gives the minimum makespan for the parallel line job shop scheduling and the sequence corresponding to the minimum makespan. The results obtained by the proposed algorithms are given in Table 8. The difference in time units is 14 in between GA and PSO. The results reveals that the Particle Swarm Optimisation (PSO) techniques provides best optimal solution to select the job sequence for reducing the Makespan by considering the quantity of jobs for parallel line job shop scheduling.

REFERENCES

Bean, J.C., 1994. Genetic algorithms and random keys for sequencing and optimization. ORSA J. Comput., 6(2): 154-160.

Behnamian, J., 2014. Particle swarm optimization-based algorithm for fuzzy parallel machine scheduling. Int. J. Adv. Manuf. Tech., 75(5): 883-895.

Cheng, B., Q. Wang, S. Yang and X. Hu, 2013. An improved ant colony optimization for scheduling identical parallel batching machines with arbitrary job sizes. Appl. Soft Comput., 13(2): 765-772.

Elmi, A., M. Solimanpur, S. Topaloglu and A. Elmi, 2011. A simulated annealing algorithm for the job shop cell scheduling problem with intercellular moves and reentrant parts. Comput. Ind. Eng., 61(1): 171-178.

French, S., 1982. Sequencing and Scheduling: An Introduction to the Mathematics of the Job-Shop. John Willey and Sons Inc., New York, USA.

Geyik, F. and I.H. Cedimoglu, 2004. The strategies and parameters of tabu search for job-shop scheduling. J. Intell. Manuf., 15(4): 439-448.

Gromicho, J.A.S., J.J. Van Hoorn, F. Saldanha-da-Gama and G.T. Timmer, 2012. Solving the job-shop scheduling problem optimally by dynamic programming. Comput. Oper. Res., 39(12): 2968-2977.

Haq, A.N., K. Balasubramanian, B. Sashidharan and R.B. Karthick, 2008. Parallel line job shop scheduling using genetic algorithm. Int. J. Adv. Manuf. Tech., 35(9): 1047-1052.

Musharavati, F. and A.S.M. Hamouda, 2011. Modified genetic algorithms for manufacturing process planning in multiple parts manufacturing lines. Expert Syst. Appl., 38(9): 10770-10779.

Panneerselvam, R., 2012. Production and Operations Management. 3rd Edn., PHI Learning Pvt. Ltd., New Delhi.

Solimanpur, M. and A. Elmi, 2013. A tabu search approach for cell scheduling problem with makespan criterion. Int. J. Prod. Econ., 141(2): 639-645.

Tang, D. and M. Dai, 2015. Energy-efficient approach to minimizing the energy consumption in an extended job-shop scheduling problem. Chin. J. Mech. Eng., 28(5): 1048-1055.

Tasgetiren, M.F., Y.C. Liang, M. Sevkli and G. Gencyilmaz, 2007. A particle swarm optimization algorithm for makespan and total flowtime minimization in the permutation flowshop sequencing problem. Eur. J. Oper. Res., 177(3): 1930-1947.

Zhang, L., L. Gao and X. Li, 2013. A hybrid intelligent algorithm and rescheduling technique for job shop scheduling problems with disruptions. Int. J. Adv. Manuf. Tech., 65(5): 1141-1156.