Asian Journal of Information Technology 15 (13): 2210-2216, 2016
ISSN: 1682-3915

An Efficient Algorithm for Job Scheduling Problem-Enhanced Artificial Bee Colony Algorithm

V. Santhi and S. Nandhini
Department of Computer Science and Engineering,
PSG College of Technology, Anna University, Chennai, India

Abstract: Job scheduling algorithm is used for assigning the jobs in number of machines that will optimize the overall performance of the running application. In this study, we proposed enhanced Artificial Bee Colony (ABC) algorithm with cross over and mutation operator for job scheduling. The main objective of this algorithm is to obtain a best schedule for jobs which minimizes the makespan value. The processing time of the jobs are generated randomly by using normal, uniform and exponential distribution. The best schedule obtained is then compared with schedule obtained from normal ABC algorithm. The computational results show that the enhanced ABC proves to be a better algorithm than the normal ABC algorithm.

Key words: Job scheduling, optimization, artificial bee colony, Makespan minimization, value

INTRODUCTION

Today world is online application specific. There are number of jobs or processes running in a distributed environment. They need more number of resources over time for completion of their tasks. Hence, the scheduling is the most important issues among all running jobs. Scheduling is the process of allocating resources to the jobs or processes in efficient manner. The job or process scheduling is one of the most important optimization problems. The problem is more complex and is proved to be NP hard problem. Here finding an optimized schedule plays an important role that leads to provide minimized makespan value. Many researchers have been worked on this job scheduling problem. Due to their computational complexity the job scheduling cannot be solved by exact algorithms. Hence many researchers have given different solutions by using heuristic and metaheuristic approach. The ultimate aim of these algorithms is to minimize the makespan value or total flow time value of individual job that are running on different resources. There are number of evolutionary algorithms used for providing the solution to minimize the makespan value. ABC (Karaboga and Basturk, 2007, 2008; Karaboga and Akay, 2009; Zhang and Wu, 2011) is one of such algorithm to provide the solution for this.

Most of the ABC scheduling algorithms are deterministic in which the processing time of all jobs are well known in advance and also it is fixed. But in real world, the processing time of jobs are dynamic in nature. It is affected by uncertain parameters. To handle this situation, the enhanced ABC algorithm is proposed. This algorithm works on stochastic environments using random processing time of jobs with known probability distributions. This algorithm can also take due dates of individual jobs. The due dates are either fixed or random based on the nature of jobs. The main objective of this algorithm is to find a feasible schedule that lead to minimize the completion time of all the jobs running on the system.

Literature review: Golenko and Gonik (2002) developed an optimal job-shop scheduling. Here there are several decision making rules used for selecting best job among number of jobs waiting for a particular machine. Making the rules are tedious here. Tavakolli-Moghaddam *et al.* (2005) developed a hybrid method for solving stochastic job shop scheduling by using both neural networks and simulated annealing. Azadeh *et al.* (2012) developed a hybrid computer simulation-artificial neural network algorithm for optimization of dispatching rule selection in job-shop scheduling. Michael Andresen developed scheduling algorithm using simulated annealing for n jobs and n machines. It was open shop scheduling with known release date of job, job weight and a due date. Recently there are number of job shop scheduling algorithms with different optimization functions developed based on genetic algorithms (Pezzella *et al.*, 2008; Lei, 2011).

Corresponding Author: V. Santhi, Department of Computer Science and Engineering, PSG College of Technology, Anna University, Chennai, India

But these algorithms are time consuming. Pan *et al.* (2011) developed an ABC for a flow-shop scheduling problem with enhanced version of the normal ABC. But it works under discrete environment. Banharnsakun *et al.* (2012) developed a scheduling algorithm based on best-so-far solution rather than a neighboring solution as proposed in the normal ABC method. Tasgetiren *et al.* (2011) developed an ABC algorithm in discrete nature for solving scheduling problem. The algorithm gives a schedule of n items in cyclic manner on a particular machine. Pansuwan *et al.* (2010) proposed an ABC algorithm for minimizing both earliness and tardiness cost with help of just in time philosophy. Ziarati *et al.* (2011) proposed an algorithm in which the activities are selected based on their ranks and the priority rules are used for ranking the activities.

MATERIALS AND METHODS

System design: The proposed system consists of m machines and n jobs. The system is said to complete its work only when all the jobs run on all the machines. The input to the system is the processing time in which every job run on every machine and the order of precedence of jobs running on the machine. The output of the system is the schedule which makes the make span value efficient. The input is in the form of matrices that shows the processing time and precedence of different jobs. The processing time of individual job is represented as the following input matrix:

$$\begin{pmatrix} p_{11} & p_{12} & p_{13} & p_{14}\cdots p_{1m} \\ p_{21} & p_{22} & p_{23} & p_{24}\cdots p_{2m} \\ p_{31} & p_{32} & p_{33} & p_{34}\cdots p_{3m} \\ \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots \\ p_{n1} & p_{n2} & p_{n3} & p_{n4}\cdots p_{nm} \end{pmatrix}$$

where, p_{ij} is the processing time of i_{th} job and j_{th} machine. For simplicity purpose the precedence and output matrices are represented as 3×3 instance. The precedence matrix represented as the job has a fixed path which goes across all the machines in a determined order. It takes the following form:

$$\begin{pmatrix} p_{11}\cdots> & p_{13}\cdots> & p_{12} \\ p_{23}\cdots> & p_{21}\cdots> & p_{22} \\ p_{32}\cdots> & p_{31}\cdots> & p_{33} \end{pmatrix}$$

Where:

P_{ij} = Denotes precedence

...> = Denotes job i must run on machine j only after i_{th} job run on $(j-1)^{th}$ machine

The output matrix is represented as:

$$\begin{pmatrix} j_{11} & j_{12} & j_{13} \\ j_{21} & j_{22} & j_{23} \\ j_{31} & j_{32} & j_{33} \end{pmatrix}$$

where, j_{ij} represents j_{th} job run on i_{th} machine. It shows the efficient schedule of the system. The work of ABC and enhanced ABC algorithm is to schedule in such a way that it minimizes the overall makespan value. For this many assumptions are made such as:

- The processing time must not be zero
- The output must not change the precedence order given as the input
- There are all machines in working condition
- The jobs are non-pre emptive
- Each machine can take only one job at a time

The number of solutions generated both by the ABC and enhanced ABC algorithm are given as the random inputs.

Algorithm implementation: Here the system consists of m machines and n jobs. The enhanced ABC and ABC are applied to schedule the jobs on the machines in an efficient way such that it reduces the total makespan value required for all the jobs to complete their operation. The output of this system is the matrix which is represented by a of size m×n in which each row represents the machines and value in the matrix represents the jobs. This matrix gives the efficient schedule of each machine. The block diagram of the overall system is represented in Fig. 1. ABC consists of three phases as follows:

- Initialization phase
- Employed bee phase
- Onlooker bee phase

Whereas, enhanced ABC consists of five phases as follows:

- Initialization phase
- Employed bee phase
- Crossover phase
- Onlooker bee phase
- Mutation phase

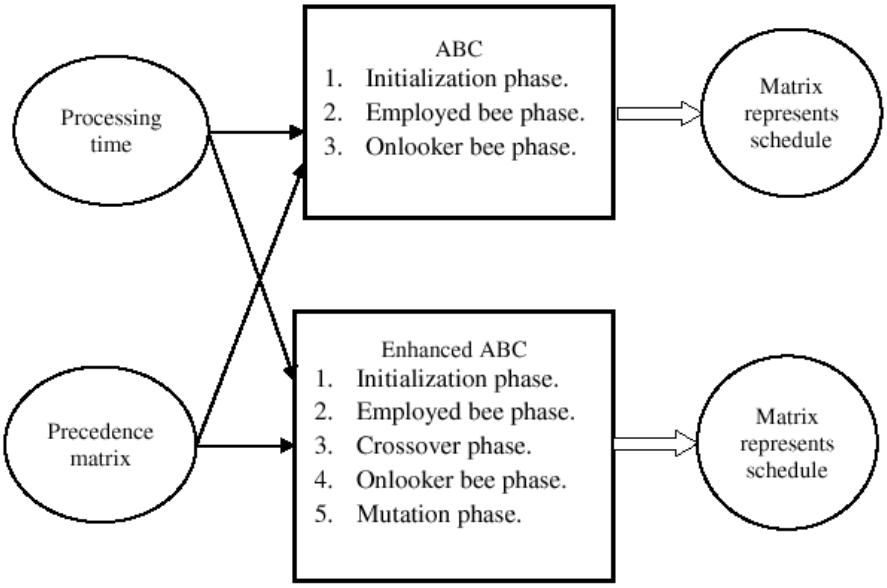

Fig. 1: Block diagram of the overall system

Initialization phase: This phase has the following steps:

- Create m queues and calculate length which is equal to m×n
- Find what are the jobs waiting for each machine and add them in the particular queue
- Pre compute the following terms for the jobs waiting in the queue to use them in Apparent Tardiness Cost (ATC) (Vepsalainen and Morton, 1987) rule. This rule is used to find the best fit job ($B_{ij}(t)$) to be scheduled among the jobs waiting in the queue when a machine is freed at time t
- The set of job successor for the current job denoted by $JS(j_{ij})$
- Average processing time of currently waiting jobs in that particular machine's buffer denoted by $\bar{p}$
- Estimated lead time of current job denoted by W_{ij}
- According to ATC rule, $B_{ij}(t)$ is calculated as follows

$$B_{ij}(t) = \frac{w_j}{p_{ij}} . \exp\left\{ - \frac{\left[d_j - t - p_{ij} - \sum_{JS(j_{ij})} \left(W_{ij} + p_{ij} \right) \right]}{K.\bar{p}} \right\} \quad (1)$$

Where:
w_j = The waiting time of that particular job
p_{ij} = The processing time of the current job
d_j = The level of urgency for the job j
k = The scaling factor which is assumed to be 2

- Reduce the value of length by 1 after each job gets scheduled. Continue above steps until the value of length becomes zero
- At the end of this a solution will be obtained for the given problem

Repeat the above steps employed_bee times to compute number of different solutions for the same problem.

Employed bee phase: This phase has the following steps:

- Neighbourhood search is done on solution by randomly applying changes on that solution
- Compute the fitness (makespan) value for the newly computed solution
- If the fitness value of the new solution is better than the existing solution then, replace the old solution by newly computed solution
- Repeat the above steps for all solutions computed in the initialization phase

Crossover phase: This phase has the following steps:

- Cross two solutions and generate a new solution called offspring
- Calculate the fitness value for the new solution.
- If any of the parent's solution is worst than this solution then, replace worst parent with the offspring
- Repeat the above steps for all pair of solutions

Onlooker bee phase: This phase has the following steps:

- Among all the solutions resulted from crossover bee phase choose the best solution based on the probability value computed as follows

$$p_i = \frac{f_i}{\sum_{i=1}^{SN} f_i} \quad (2)$$

Where:

f_i = The fitness of the solution

I and SN = The number of solutions

Mutation phase: This phase has the following steps:

- Uniform mutation is applied on the solution resulted from the onlooker bee phase to obtain new solution
- Fitness value is calculated for the new solution
- If the mutation increases the fitness value then replace the existing solution

The pseudo code of the proposed algorithm:

Step 1: Intialization phase:
for I = 0 to number of solution do
length = no_of_machines*no_of_jobs
while length >0 do
for j = 0 to number of machines
Allocate job to the machine j at time t length
end
end
Calculate makespan value for the current solutions
end
Step 2: Employed bee phase:
for I = 0 to number of solution do
Conduct neighbourhood search for the current solution
Calculate makespan value for the new solution
If new makespan value is efficient than the current makespan
then replace the existing solution with the new solution
end
end
Step 3: Crossover phase:
for I = 0 to number of solution/2 do
Combine pair of existing solutions to create new solution called offspring
Calculate makespan value for offspring
Replace the worst parent with the offspring if it is better than that
end
Step 4: Onlooker bee phase:
for I = 0 to number of solution do
Calculate probability value for the current solution
end
Select the solution with greater probability as the result of onlooker bee phase
Step 5: Mutation phase:
Apply mutation operator to the resultant solution from above phase
Calculate makespan value for new solution
if new makespan is efficient than existing
then
Replace the existing solution with the new solution.
end for if

Finally, the solution obtained from the mutation bee phase is considered as the ultimate solution given by the enhanced ABC algorithm. The flowchart of this algorithm is represented in Fig. 2.

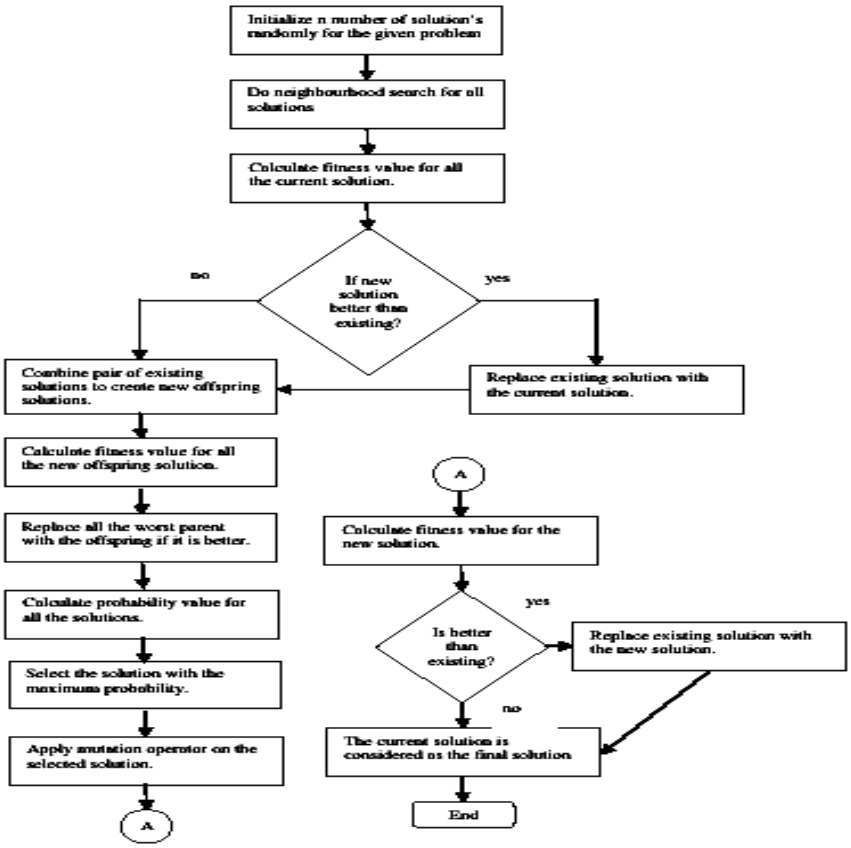

Fig. 2: Enhanced ABC algorithm

Table 1: The computational result of m machines and n jobs under normal distribution with θ = 0.1

		ABC			Enhanced ABC		
Size m×n	Instance number	Best	Average	Worst	Best	Average	Worst
10×10	1	51.251	65.625	75.402	45.144	53.673	61.692
	2	40.939	46.107	51.508	38.625	43.481	48.934
	3	52.089	63.981	69.117	52.081	56.753	63.930
	4	50.962	59.861	60.582	48.562	54.711	59.321
	5	56.120	61.354	67.508	50.457	54.893	59.642
15×10	6	65.399	69.752	72.852	64.306	67.973	70.743
	7	92.016	95.457	98.895	90.231	92.673	93.989
	8	76.712	80.836	83.947	31.264	34.946	38.932
	9	67.644	68.972	70.953	32.768	36.826	40.971
	10	98.319	102.450	108.445	93.219	95.314	98.937
15×15	11	93.923	97.456	101.963	93.310	94.982	96.346
	12	86.791	90.347	92.098	83.791	85.360	89.861
	13	104.421	106.027	109.837	104.421	105.221	107.582
	14	61.483	64.862	68.852	58.356	60.349	63.495
	15	102.615	105.387	109.546	101.093	103.349	106.462
20×20	16	132.595	138.986	143.863	94.183	98.825	103.452
	17	156.231	160.386	167.384	144.876	147.863	151.341
	18	92.457	95.954	98.620	90.659	93.865	97.644
	19	100.230	103.086	107.393	98.003	101.245	105.781
	20	154.823	159.041	162.428	148.907	150.855	154.936

RESULTS AND DISCUSSION

The performance analysis of ABC and enhanced ABC is done by, comparing the results generated by both the algorithms provided the same set of processing time are given to them. The processing time are randomly generated for three different distributions namely: normal distribution, Uniform distribution and exponential distribution. In all test cases, we consider m machines and n jobs with instances.

The instances are indexed with i and j. Here i refers job instance at time t and j refers machine instance at time t. In each instance, the path is a random precedence of m machines. The common computational time is set for both ABC and enhanced ABC algorithm. The common computational time is 50 sec. The best, average and worst makespan values are taken from random generated processing times.

Normal distribution: The processing time under normal distribution is generated based on the following: for job i and machine j, the processing time p_{ij} is calculated as follows:

$$p_{ij} = N(mean_{ij}, std_{ij}) \quad (3)$$

Where:

N = Normal distribution

$mean_{ij}$ = Generated from uniform distribution within the interval (1, 99) and standard deviation

std_{ij} = Derived from Eq. 4

$$std_{ij} = \Theta \times mean_{ij} \quad (4)$$

where, Θ is level of variability. Table 1 shows the computational result of m machines and n jobs under normal distribution with Θ = 0.1.

Uniform distribution: The uniform distribution represents a situation where all outcomes in a range between a minimum and maximum value since every outcome is equally likely to occur. The processing time under uniform distribution is generated based on the following:

$$p_{ij} = U(mean_{ij} - wid_{ij}, mean_{ij} + wid_{ij}) \quad (5)$$

Where:

U = Uniform distribution and width parameter

wid_{ij} = Derived from Eq. 6

$$wid_{ij} = \Theta \times mean_{ij} \quad (6)$$

where, Θ is level of variability. Table 2 shows the computational result of m machines and n jobs under uniform distribution with Θ = 0.1.

Exponential distribution: Exponential distribution will represents the time between events in a poisson process. The processing time under exponential distribution is generated based on the following:

$$p_{ij} = EXP(\lambda_{ij}) \quad (7)$$

where, EXP is exponential distribution and:

$$\lambda_{ij} = 1/mean_{ij} \quad (8)$$

Table 2: The computational result of m machines and n jobs under uniform distribution with Θ = 0.1

		ABC			Enhanced ABC		
Size m×n	Instance number	Best	Average	Worst	Best	Average	Worst
10×10	1	52.088	56.753	69.113	42.512	52.241	63.191
	2	51.251	61.108	73.701	33.359	45.144	51.108
	3	50.962	54.106	59.861	43.062	48.244	51.321
	4	56.120	60.410	63.771	34.431	39.737	46.862
	5	40.939	48.549	53.634	26.436	33.438	40.931
15×10	6	65.398	69.753	72.982	38.727	43.852	51.369
	7	92.016	95.378	99.564	66.918	70.572	75.954
	8	76.712	80.644	84.874	24.912	31.738	42.854
	9	67.644	73.843	81.845	19.687	27.746	31.874
	10	93.219	102.747	113.685	59.482	64.854	71.758
15×15	11	93.310	95.358	97.978	88.858	90.945	94.345
	12	86.790	89.304	93.683	43.755	49.435	57.564
	13	104.426	108.453	116.435	92.047	97.987	99.430
	14	61.483	64.653	69.456	61.433	64.563	68.657
	15	102.505	106.566	110.345	91.680	94.546	96.950
20×20	16	158.394	163.873	171.558	114.916	123.641	132.743
	17	132.595	138.673	143.782	83.291	89.578	94.952
	18	156.238	160.742	163.784	144.876	150.546	154.742
	19	92.360	98.742	103.239	85.700	92.785	97.238
	20	150.427	154.734	160.845	110.564	116.874	121.032

Table 3: The computational result of m machines and n jobs under exponential distribution

		ABC			Enhanced ABC		
Size m×n	Instance number	Best	Average	Worst	Best	Average	Worst
10×10	1	92.123	97.248	102.258	87.232	89.362	99.456
	2	94.432	98.482	104.236	90.324	93.982	99.237
	3	95.137	96.457	107.567	89.460	92.342	102.287
	4	110.614	114.736	132.496	103.159	109.351	128.143
	5	104.349	107.217	128.320	98.353	101.235	123.640
15×10	6	112.342	114.287	130.187	105.643	110.153	124.364
	7	121.356	127.456	140.753	109.349	116.430	132.730
	8	118.157	124.287	139.563	106.237	119.415	123.437
	9	103.262	107.454	127.361	96.325	102.737	122.173
	10	117.224	119.856	132.452	112.768	112.613	128.281
15×15	11	120.413	122.316	139.124	110.235	117.219	130.126
	12	123.523	125.213	148.234	117.642	121.342	141.579
	13	142.349	144.453	164.543	134.321	138.548	161.328
	14	129.743	131.238	159.438	121.467	126.634	153.129
	15	137.294	139.314	156.312	126.238	132.416	149.253
20×20	16	167.423	169.246	203.125	148.234	149.453	176.237
	17	158.157	161.458	192.453	139.237	142.103	182.423
	18	176.234	180.242	210.463	157.234	158.543	198.231
	19	185.473	191.547	220.821	164.345	167.249	204.135
	20	198.364	201.261	227.632	180.275	182.347	218.127

Table 3 shows the computational result of m machines and n jobs under exponential distribution. Based on the results shown in all tables, it can be observed that the total makespan value using enhanced ABC is lesser than the total makespan value using ABC algorithm under different distributions and under different dimensions. So, it can be concluded that using enhanced ABC, the completion time for each job under each machine can be further reduced thus leading to an optimal solution.

In addition, the Mann-Whitney U-test is conducted for comparing the values in all tables statistically. It is non-parametric test. Here the ABC and enhanced ABC algorithms are run for 15 independent times on first ten instances. Then n1 = n2 = 15 in the Mann-Whitney U-test. The critical value of U is 90 for two-tailed test of Mann-Whitney U-test at the 0.05 significance level. The critical value of U is 73 for two-tailed test of Mann-Whitney U-test at the 0.01 significance level. To be significant, the obtained U for first ten instances has to be less than the available critical value of U. The obtained U value is shown in Table 4 based on ten instances. So, the obtained U is less than critical value U for both 0.05 and 0.01 significance level. Based on statistical evaluation, the enhanced ABC is significantly better than the normal ABC algorithm.

Table 4: Obtained u-value from mann-whitney u tests on the computational results

Size m×n	Instance number	Normal distribution with Θ = 0.1	Uniform distribution with Θ = 0.1	Exponential distribution
10×10	1	48	49	31
	2	43	44	27
	3	45	44	28
	4	42	43	24
	5	49	40	20
15×10	6	46	49	18
	7	42	45	17
	8	49	42	17
	9	46	40	15
	10	43	39	13

CONCLUSION

In this study, an enhanced ABC is proposed for getting the best schedule in scheduling system with randomly generated processing times of individual jobs. The computational results under different probability distributions show that the makespan value of enhanced ABC is better than the normal ABC. The best schedule is used for minimizing the makespan value. Our future research lies in experimenting and making use of the many new evolutionary algorithms that have been proposed to improve the performance of the job scheduling algorithm in a distributed systems environment.

REFERENCES

Azadeh, A., A. Negahban and M. Moghaddam, 2012. A hybrid computer simulation-artificial neural network algorithm for optimisation of dispatching rule selection in stochastic job shop scheduling problems. Int. J. Prod. Res., 50: 551-566.

Banharnsakun, A., B. Sirinaovakul and T. Achalakul, 2012. Job shop scheduling with the best-so-far ABC. Eng. Appl. Artif. Intell., 25: 583-593.

Golenko G.D. and A. Gonik, 2002. Optimal job-shop scheduling with random operations and cost objectives. Int. J. Prod. Econ., 76: 147-157.

Karaboga, D. and B. Akay, 2009. A comparative study of artificial bee colony algorithm. Appl. Math. Comput., 214: 108-132.

Karaboga, D. and B. Basturk, 2007. A powerful andefficient algorithm for numerical function optimization: Artificial Bee Colony (ABC) algorithm. J. Global Optim., 39: 459-471.

Karaboga, D. and B. Basturk, 2008. On the performance of Artificial Bee Colony (ABC) algorithm. Appl. Soft Comput., 8: 687-697.

Lei, D., 2011. Simplified multi-objective genetic algorithmsfor stochastic job shop scheduling. Appl. Soft Comput., 11: 4991-4996.

Pan, Q.K., M.F. Tasgetiren, P. Suganthan and T.J. Chua, 2011. A discrete artificial bee colony algorithm for the lot-streaming flow shop scheduling problem. Inform. Sci., 181: 2455-2468.

Pansuwan, P., N. Rukwong and P. Pongcharoen, 2010. Identifying optimum Artificial Bee Colony (ABC) algorithm's parameters for scheduling the manufacture and assembly of complex products. Proceedings of the 2010 Second International Conference on Computer and Network Technology (ICCNT), April 23-25, 2010, IEEE, Bangkok, Thailand, ISBN: 978-0-7695-4042-9, pp: 339-343.

Pezzella, F., G. Morganti and G. Ciaschetti, 2008. A genetic algorithm for the flexible job-shop scheduling problem. Comput. Operat. Res., 35: 3202-3212.

Tasgetiren, M.F., O. Bulut and M.M. Fadiloglu, 2011. A discrete artificial bee colony algorithm for the economic lot scheduling problem. Proceedings of the 2011 IEEE Congress of Evolutionary Computation (CEC), June 5-8, 2011, IEEE, New Orleans, Louisiana, ISBN: 978-1-4244-7834-7, pp: 347-353.

Tavakolli-Moghaddam, R., F. Jolai, F. Vaziri, P.K. Ahmed and A. Azaron, 2005. A hybrid method for solving stochastic job shop scheduling problems. Applied Mathe. Comput., 170: 185-206.

Vepsalainen, A.P.J. and T.E. Morton, 1987. Priority rules for job shops with weighted tardiness costs. Manage. Sci., 33: 1035-1047.

Zhang, R. and C. Wu, 2011. An artificial bee colony algorithm for the job shop scheduling problem with random processing times. Entropy, 13: 1708-1729.

Ziarati, K., R. Akbari and V. Zeighami, 2011. On the performance of bee algorithms for resource-constrained project scheduling problem. Applied Soft Comput., 11: 3720-3733.