Hybridization of Evolutionary and Swarm Intelligence Techniques for Job Scheduling Problem

V.Selvi[#1], R. Uma Rani*[2], R.Sankar[#3]
[#]Department of M.C.A,
M.A.M.College of Engineering,Siruganur,Trichy,Tamilnadu,India.
Department of Computer Science,
Shri saratha College for women, Salem, Tamilnadu. India.
Selvigiri.s@gmail.com
Sankargenuine@gmail.com
[*]umainweb@gmail.com

Abstract- For more than a decade, scheduling of jobs has been an attractive research subject for researchers. There are several different ways to schedule jobs, and the threads which make them up. As well, the job scheduling is one of the active research fields, where the researchers work to enhance the efficiency of the job scheduling process in a scheduling environment. In existing hybrid techniques, some efficient factors related to jobs like turnaround time, job execution time and more have not been considered in the job scheduling process. The main drawback is lack of factors in the scheduling process which reduces the performance. To overcome such drawback in the existing methods, an adaptive ABC technique is proposed. In this proposed adaptive ABC technique, the term adaptiveness is achieved by using mutation, crossover and velocity in the employed bee phase for finding the new food sources. The adaptive ABC algorithm optimally allocates the jobs to the accurate processors or resources. Moreover, these existing techniques mostly concentrate on two major factors such as the minimization of the makespan and the completion time. The adaptiveness improves the efficiency of scheduling process when compared to the two conventional hybrid job scheduling techniques. The experimental result shows the performance of the proposed job scheduling process.

Keywords: Job Scheduling, Genetic Algorithm (GA), Ant Colony Optimization (ACO), Particle Swarm Optimization (PSO), Artificial Bee Colony Optimization (ABC).

I. INTRDUCTION

An Evolutionary swarm intelligence technique is applied in this paper for solving job scheduling problem. It is a combinatorial optimization problem. Where the jobs are arranged, so that they may or may not processed in every machine sequence. Then each machine shall have a unique sequencing of jobs. An extended version of flow scheduling is known as job scheduling. The main objective of both kinds of problem is jobs in a sequence that gives a minimum value of makespan. Over the long time scientist have developed population based algorithm and the most common algorithms are genetic algorithm, ant colony optimization algorithm and Particle swarm optimization [1].

An algorithm which is developed for the travelling salesman problem is an ant colony optimization. It is an ant communicative behaviour where ants communicate with each other through pheromone, which is kind of chemicals substance in insects. This behaviour of ants helps us to find the shortest path from their nest and food source. In this paper job scheduling problem with the makespan objectives solved using ant colony optimization. The next most common population based algorithm is particle swarm optimization, which shares many similarities with genetic algorithms. Random selections are undergone for system initialization. There is no cross over mutation for genetic algorithm and Particle swarm optimization all. Combination of job sequence can be found using particle swarm optimization. Then the algorithm is extended after understanding particle swarm optimization algorithm and is applied in job scheduling problem for n jobs and m machines.

Efficiency is one in job scheduling which plays in important role, if jobs and machines are considered in large numbers [2]. Decoding schedule in the algorithm is done using operation based representation. Inverse and pair wise mutations are the two mutations which are used in the algorithm.Time and operation sequence are the two input parameters which are offered. Then the result is compared. Set of jobs and machines are there and each job consists of chain and is processed during an uninterrupted time period of that particular length on a given machine[5]. Only one process can be done at a time. The main aim is to find the schedule of minimum length where a schedule is an allocation of operations to time intervals of the machines.

In literature, different hybrid techniques are applied for performing the job scheduling process in various fields. The existing systems have applied techniques such as Artificial Bee Colony (ABC), Genetic Algorithm

(GA), Particle Swarm Optimization (PSO), Ant Colony Optimization (ACO), Simulated Annealing (SA) etc. Hybrid techniques were derived from the combination of these algorithms. These techniques, mostly concentrate on two factors such as **(i)** the minimization of the makespan and **(ii)** the completion time. The hybrid techniques have performed well with these two factors and allocated jobs to the resources for increasing the resource utilization. However, the operations of these hybrid techniques lack in the scheduling process because, such techniques have not considered the efficient factors like (i) turnaround time, (ii) throughput, (iii) response time, and so forth during the job scheduling process. The performance has been reduced due to lack of such factors in the scheduling process.

In this paper, the methods of Genetic Algorithm (GA), Ant Colony Optimization (ACO), Particle swarm optimization (PSO) and Artificial Bee Colony (ABC), have been studied and is being applied to the job scheduling case so as to get an optimum processing time. Many of the recent research studies focus on metaheuristic approaches such as ACO_PSO, PSO_GA and ABC_GA with job scheduling. The above hybrid techniques are completed with minimum make span and completion time. In this paper an Adaptive Artificial Bees Colony (AABC) algorithm is used to solve JSP which can follow with crossover and mutation operators of GA and the rest is with a velocity of PSO. This is also hybrid technique which is better than the above existing algorithm (i.e.) makespan and completion time is minimized when compared to existing algorithm.

The rest of this report is coordinated as follows: Section 2 briefly reviews the genetic algorithm. Section 3 briefly describes about the Swarm intelligence techniques. Section 4 describes about Adaptive ABC for Job Scheduling Problem. Section 5 discusses about Experimental results and discussion and finally some conclusion are made in Section 6.

II. GENETIC ALGORITHM

Evolution based on genetic processes of biological is known as genetic algorithm. It is introduced by john Holland (Goldberg 1989) and GA is categorized as global search heuristics. Some techniques which are inspired by evolutionary biology are inheritance, mutation, selection and cross over and those techniques are which are of a particular class.

Chromosomes are present in GA, Where each generation consists of a population of binary strings known as chromosomes[6]. Development takes place often and encoding of new properties in the chromosomes takes place. The highest fitness of chromosomes is being taken the next generation.

In ordination, to increase the diversity of perturbed offspring, the crossover operator enables an exchange of substrings between two parents [4].

The Selection operator decides whether or not a new solution should be selected to be a member of the next generation. Meanwhile, the random modification of a new configuration is controlled by the mutation operator. It can be used for combinatorial optimization, pattern recognition, machine learning, planning strategy and information processing. This algorithm has many advantages, such as: doing a global search quickly and well; simple process of utilizing the evaluation function to search; randomness iterated by probability mechanism; well associatively combining with other optimization techniques.

Since GA can operate along more than one result at one time, a group of initial solutions is determined at the start. The fitness value of each solution is calculated according to the objective function and new search solutions are generated randomly by examining its fitness value and applying the genetic operators (crossover operator and mutation operator) to the search solutions has better fitness value, it will replace the older one in our experiment[6]. While the best individual is carried over from one generation to the next.

Basic Genetic Algorithm

1. Generate random population of n chromosomes (suitable solutions for the problem)

2. Repeat

3. For Population size

4. Evaluate the fitness f(x) of each chromosome x in the population. Select two parents from old generation.

5. Create a new population by repeating the following steps until the new population is complete. Recombine parents for two offspring.

5.1 Select two parent chromosomes from a population according to their fitness.

5.2 Generate an Initial population .With a crossover probability crossover the parents to form a new offspring. If no crossover is performed, the offspring is an exact copy of parents. Compute fitness of each individual

5.3 Insert offspring in new generation

End for

6. With a mutation probability mutate new offspring at each locus (position in chromosome).

7. Place new offspring in a new population. Use the newly generated population for a further run of algorithm.

8. Stop and return the best solution in current population until the end condition is satisfied.

III. SWARM INTELLIGENCE

Swarm intelligence (SI) is the collective behavior of decentralized, self- formed systems, natural or artificial [10].The concept is employed in work on artificial intelligence. Swarm Intelligence principles have been successfully applied in a variety of problem domains including function optimization problems, finding optimal routes and scheduling. The main Swarm Intelligence based methods are Ant Colony Optimization (ACO), Particle Swarm Optimization (PSO), and Artificial Bee Colony Optimization algorithm.

A. Ant Colony Optimization

Ant colony optimization is a probabilistic technique which is utilized for solving computational problem and can find good paths using graphs. This algorithm constitutes some metaheurstics optimizations[13].The main aim of the first algorithm is to search for an optimal path in a graph which is based on their behavior and there was introduced by Marco Dorgo in 1992 in his PhD thesis. A more extensive class of mathematical problems is solved and as a consequence, several problems have been merged. In ACO, search for global solutions to a given optimization problem is transformed for finding the cost path in the weighted graph which is minimal[15]. Solutions built by artificial ants moving on the graph. The result, the building process is stochastic where a lot of parameters associated with graph elements, where the values are qualified at the run time by the ants. ACO has been applied to many classical problems combine as well as to a discrete optimization problem that have stochastic and dynamic computation.

First, each instantiated decision variable $Xi = vij$ is called a *solution component* and denoted by j . The set of all possible solution components is denoted by C . Then the construction graph G_C (V,E) is defined by associating the components C either with the set of vertices V or with the set of edges E.

A pheromone trail value τij is associated with each component cij .

The amount $\Delta\tau$ of pheromone deposited may depend on the quality of the solution found. Subsequent ants utilize the pheromone information as a guide towards more promising regions of the search space.

The ACO metaheuristic is:

Set parameters, initialize pheromone trails

While Termination Condition is not met

Begin

Schedule_Activities

ConstructAntSolutions()

DaemonActions()

UpdatePheromones()

End_Schedule_Activities

End While

The metaheuristic is an initialization step and three algorithmic components which are regulated by SCHEDULED-ACTIVITIES are consisted by the metaheuristic. Repetition of this construction is done until a termination criterion is met. Maximum number of iterations results in typical criteria. The three scheduled and synchronized algorithmic computations are not specified by the SCHEDULE ACTIVITIES construct

B. Particle Swarm Optimization

Particle swarm optimization is a population based stochastic optimization technique for the solution of continuous optimization problems (Kennedy and Eberhart -1995)[15]. In particle swarm optimization (PSO), a set of software agents called particles search for best solutions to a given continuous optimization problem. In practice, in the initialization phase of each particle is thrown a random initial position and an initial velocity. The position of the particle represents a solution of the problem and therefore has a value, interpreted by the objective intent. At every iteration of the algorithm, every particle moves with a speed that is a weighted summation of three components: the old velocity, a velocity component that forces the particle towards the positioning in the search space where it previously found the best result so far, and a velocity component that forces the particle towards the positioning in the search space where the neighbour particles found the best answer so far. In PSO, each single solution is a "bird" in the search space Called "particle". The operation of a particle is measured by a fitness value.

The particles fly through the problem space by falling out the current optimum particles. PSO is initialized with a group of random particles and then searches for optima by updating generations. The situation of a particle is determined by the best position visited by it. Each particle knows its best position pbest and the best position so far among the full group of particles gbest.

After to finding the two best values, the particle updates its velocity and positions with the following equation (1) and (2).

$$v[] = v[] + c1 * rand() * (pbest[] - present[]) + c2 * rand() * (gbest[] - present[]) ----(1)$$

$$present[] = present[] + v[] ----(2)$$

$v[]$ is the particle velocity, present[] is the current particle. pbest[] and gbest[] are defined as stated before rand () is a random number between (0,1). c1, c2 are learning factors. Usually c1 = c2 = 2.

The pseudo code for PSO is as follows

For each particle
  Initialize particle values
END
Do
  For each particle
    Compute the fitness value
    If the fitness value is best than pBest value
      Current value= pBest
  End
  Work out the particle velocity according the particle with the best fitness value of all the particles as the gbest
  For every particle
    Work out the particle velocity according equation ------ (1)
Change particle position according equation--(2)
  End
While Minimum error criteria or Maximum iterations are not achieved.

C. Artificial Bee Colony Optimization

Inspired by the intelligent foraging behaviours of honeybee swarm, an artificial bee colony algorithm is developed, which is a new population-based meta-heuristic approach [8][9]. In ABC algorithm, there are three kinds of foraging bees: employed bees, onlooker bees, and scout bees. The procedure that followed in the ABC algorithm.

At the first step, a randomly distributed initial population (food source positions) is generated. Provided that the nectar amount of the new one is higher than that of the previous source. The bee memorizes the new source position and forgets the old one. Otherwise, it keeps the position of the one in its memory. As in the case of the employed bee, it produces a modification on the source position in its memory and checks its nectar amount providing that its nectar is higher than that of the previous one.

The basic Artificial Bee Colony algorithm

Initialization phase

Step 1: Initialize parameters, including the number of food sources or the number of employed bees, number of onlooker bees and number of scout bees.

Step 2: Initialize population of food sources with random solutions.

Step 3: Calculate the objective value of each food source and then determine the best food resource.

Employed bee phase

Step 4: For every employed bee, generate a new food source.

Step 5: Calculate the objective value for every new food sources and compute the best food source.

Onlooker bee phase

Step 6: Calculate the probability of selecting food source using (1).

Step 7: Calculate the number of onlooker bees to be sent to the food source.

Step 8: For every onlooker bee, generate the new food sources.

Scout bee phase

Step 9: Initialize scout bees with random solutions and update the best food sources.

Step 10: Determine the worst employed bees and replace them with the scout bees if the scout bees are better.

Step 11: If a stopping criterion is met, then the best food source is obtained with its objective value.

Employed bees are those bees that are exploiting a food source currently, onlooker bees are those bees that are waiting in the hive for the data to be partaken by the employed bees about their food origins, and scout bees are those bees that are currently researching for new food sources in the locality of the hive. The working process of the bees is described as follows:

The employed bees brings loads of nectar from the food sources to the hive and then share the food source information with onlooker bees by dancing in a common area in the hive called dance area. The duration of a dance is proportional to the nectar content of the food source currently being exploited by the dancing bee. Onlooker bees need to watch numerous dances before choosing a food source, which tend to choose a food source according to the probability proportional to the quality of that food source. Therefore, the best food sources tend to attract more bees than the bad ones. A scout or onlooker bee may change into an employed bee when it finds a better source. An employed bee associated with a food source may become a scout or onlooker bee when the nutrient source is exploited to the full. Employed bees and scout bees can be visualized as performing exploration, whereas onlooker bees can be visualized as performing exploitation.

In the ABC algorithm, each food source represents a possible solution to the problem under consideration, and the nectar amount of a food source represents the quality of the solution. The ABC algorithm assumes that there is only one employed bee for every food source, i.e., the number of food sources is same as the number of employed bees. The employed bee of an abandoned food source becomes a scout bee, and as soon as it finds a new food source, it becomes an employed bee again. The ABC algorithm is an iterative algorithm [14]. It starts by associating all employed bees with randomly generated food sources (solution).

Then, every employed bee moves to a new food source in the neighbourhood of its currently associated food source and evaluates its nectar amount (objective value) during iterations. When the employed bees complete the process, they share the nectar information of the food sources with the onlooker bees, and the number of onlooker bees to be sent to the food source found by the employed bee is proportional to the nectar amount of that food source.

The probability p_i of selecting a food sources i is determined by the following expression:

$$p_i = \frac{fit_i}{\sum_{i=1}^{l} fit_i} \quad \textbf{(1)}$$

Here, $fitt_i$ is the objective value of the solution represented by the food sources i and l is the total number of food sources. Clearly, good food sources will attract more onlookers than the bad ones.

IV. ADAPTIVE ABC FOR JOB SCHEDULING PROBLEM

The adaptive ABC algorithm is initiated by generating food sources f_s. The food sources are represented as $f_{s_i} = \{R_1, R_2, \cdots R_k\}; k \in M$, where i represents the number of generated food sources. In this food source, each resource has the allocated jobs based on their execution and processing times e_n and $p_{n,m}$. The proposed adaptive ABC algorithm framework is as follows.

The generated food source representation for the proposed job scheduling problem is shown in fig.1.

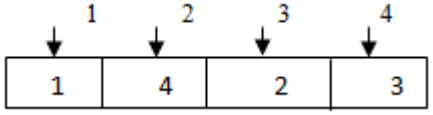

Fig. 1: Generated Food Source Representation

After that ,the fitness value is computed for the food sources. The fitness function calculation formula is given below,

$$p_i = \frac{fit_i}{\sum_{i=1}^{l} fit_i} \quad (1)$$

$$F(f_{si}) = \frac{1}{(\omega * (1/S)) + (\alpha * D)} \quad (2)$$

$$S = MAX\,(p_{j,k}) \quad (3)$$

$$D = \Sigma T(j) \quad (4)$$

$$T(j) = d_j - t_j \qquad (5)$$

In Equ. (2), S and D represents the makespan and delay time of the resources and jobs respectively. These makespan and delay times are multiplied with the constant factor values ω and α. In Equ. (3), the maximum processing time is calculated from the set of resources in the food sources and in Equ. (4), the jobs total delay time is calculated by utilizing the jobs deadline time d_j and turnaround time t_j. Based on the fitness values, the best food sources (b_s) having minimum fitness value are selected from the generated i number of food sources. Next, the new food sources are generated in employed bee phase by using the following three operations (Crossover Mutation and velocity).

Pseudo code for fitness

```
Double findFitness(ArrayList<Resource> resources)
 {
     double max=findMax(resources);
                double D=getD(resources);
                double alpha=0.5;
                double omega=0.2;
                double fitness=1/((omega*(1/max))+(alpha*D));
                   if(fitness<0)
                  fitness=(1+Math.abs(fitness));
 else
   fitness=(1/(1+fitness));
 return fitness;
  }
```

Crossover and Mutation in the employed bee phase

In the adaptive ABC algorithm, the crossover, mutation, and velocity operations are performed in generating the new food sources. A uniform crossover process is performed here to generate the new food sources, which is illustrated as follows.

Crossover

Step 1: For every employed bee, we select one best food source from b_s as u' and select the employed bee solution from the best population $u''(u'', u' \in b_s \text{ and } u'' \neq u')$.

Step 2: Uniformly generate a binary string i.e., 0 and 1 with the same length of the food source. Then, generate new food source by placing the element of u'' at position with bit 1 and placing the element of u' at position with bit 0.

Pseudo code for crossover

```
for(int a=0;a<bstring.length;a++)
                {
                  if(bstring[a]==0)
                     newfoodsource.add(bestfoodsource.resources.get(a));
          else
                     newfoodsource.add(bestpopulation.resources.get(a));
                }
```

Mutation

An integer r is randomly generated from 1 to N , in mutation, where N is the total number of jobs. Based on the generated integer value, r position is selected from the food sources. For each selected position, replace the resources with a randomly chosen different resource from the candidate resource set. The generated new food sources from the crossover and mutation operations are $e^1(b_s)$ then, the fitness value is calculated for these new food sources by utilizing the formula described in Equ. (2).

Pseudo code for mutation

```
int pos=r.nextInt(newfoodsource.size());
        for(int x=0;x<pos;x++)
        newfoodsource.add(newfoodsource.get(x));
```

Velocity in the employed bee phase

In PSO optimization algorithm, a velocity value is utilized for generating the new particles. In employed bee phase, the same velocity formula is exploited for generating the new food sources. Based on the velocity value, the new food sources are generated, which is represented as $e^2(b_s)$ .Subsequently, the fitness value for this $e^2(b_s)$ food source is computed by exploiting (2). In employed bee phase, the new food sources are $e^1(b_s)$ and $e^2(b_s)$. From this set of food sources, some of the best food sources having minimum fitness value are selected. The selected food sources from employed bee phase are $e(b_s)$ similarly in the onlooker and scout bee phase, the new food sources are generated based on the aforementioned crossover, mutation, and velocity operations. The new best food sources are computed by the fitness function values and these selected food sources are represented as $o(b_s)$ and $s(b_s)$. Among the food sources obtained from these three phases, select one best food source having minimum fitness value, which is denoted as P and the remaining food sources are given to the next iteration. This process is repeated until it reaches the maximum number of iterations I . The obtained best food sources are the optimal solutions for the jobs.

Pseudo code for velocity

```
double value= (wmax*velocity[a][b])+((pbest.resources.get(b).getJob().name-
bestpopulationsource.get(a).resources.get(b).getJob().name)*c1*r.nextDouble())+
((gbest.getResources().get(b).getJob().name - bestpopulationsource. get(a).resources.get(b) .
getName())*c2*r.nextDouble());
velocity[a][b]=velocity[a][b]+value;
```

V. EXPERIMENTAL RESULTS AND DISCUSSION

A. Implementation and Results

In this paper, the result obtained by the proposed algorithm is analyzed. To test the efficiency of the algorithm results of AABC is compared with results of the three hybrid algorithms viz., ACO_PSO, PSO_GA and ABC_GA. The experiments by varying the number of resources as well as varying the number of jobs are conducted and then the results with that of three existing hybrid algorithms are compared. The comparative results are shown in the following Table I and Fig. 2 to 4.

Table I: Mean Makespan time for 10 Jobs with 50,100,150,200 and 250 resources for ACO-PSO, PSO-GA, ABC-GA and AABC algorithm.

Iterations	Resources	ACO_PSO	PSO-GA	ABC-GA	AABC
50	50	32.7	32.3	34	28.3
	100	34	38	37.6	26
	150	24	27.9	32	22.6
	200	25.8	29.1	26.4	23.3
	250	24.3	31.7	27.5	23.2
100	50	33.4	34.2	35.6	28.3
	100	31.3	36.2	31.8	29.1
	150	29	27	26.7	25.2
	200	31.3	33.4	37.2	24
	250	32.5	29.2	31	27.3
200	50	27.3	21.8	24.3	21.2
	100	23.5	22.6	27.3	18.7
	150	19.7	18	21	16.3
	200	21.3	19.6	22.5	17
	250	25.6	21.6	23.6	19.2

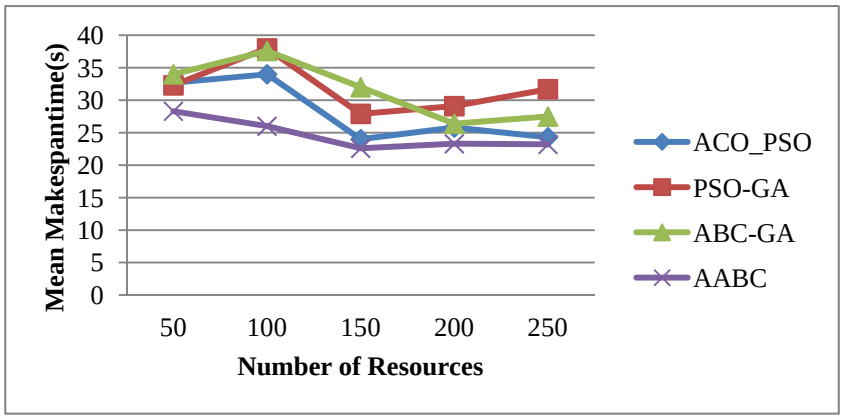

Fig.2. Mean Makespan Time for 10 jobs with 50 iterations for ACO_PSO, PSO_GA , ABC_GA and AABC.

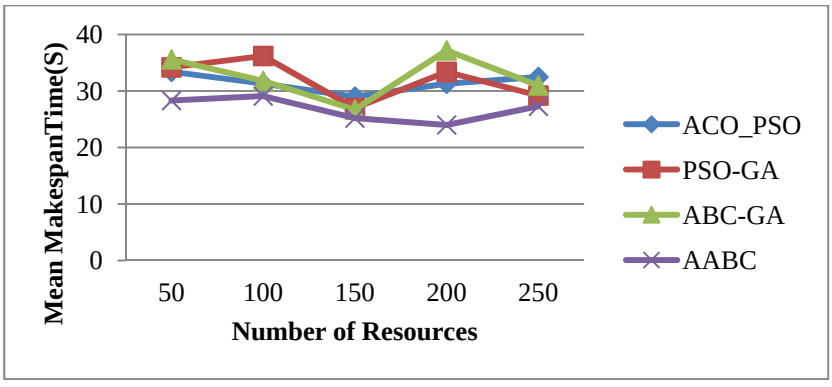

Fig.3. Mean Makespan Time for 10 jobs with 100 iterations, ACO_PSO, PSO_GA, ABC_GA and AABC.

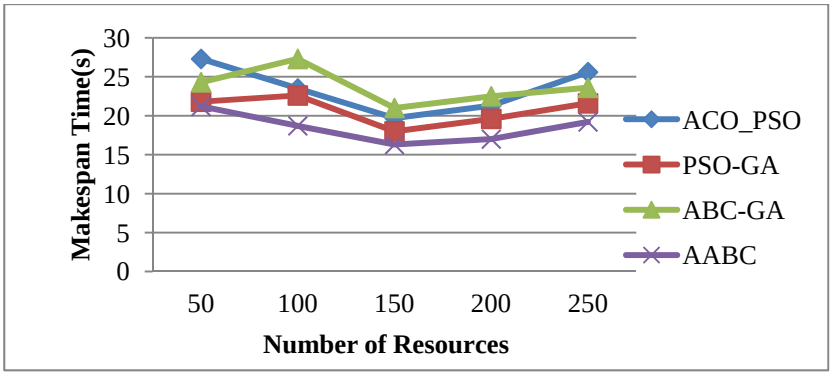

Fig.4. Mean Makespan time for 10 Jobs with 200 iterations for ACO_PSO, PSO_GA and ABC_GA, and AABC.

The various hybrid swarm intelligence algorithms (ACO_PSO, PSO_GA and ABC_GA) were compared with the proposed AABC algorithm and the performance of makespan and completion time for solving job scheduling problem were analyzed. The comparative results of mean completion time results are shown in the following Table II and fig 5 to 7.

TABLE II: Mean Completion time for 10 Jobs with 50,100,150,200 and 250 resources for PSO-CO, PSO-GA, ABC-GA, AABC algorithm.

Iterations	Resources	ACO-PSO	PSO-GA	ABC-GA	AABC
50	50	249.8	151.2	160.3	141.8
	100	196.5	237.3	224.9	191.5
	150	116.2	110.5	96.2	85.6
	200	128	117.6	110.9	97.3
	250	196.3	188.9	199.3	190.8
100	50	253.5	266.5	161.9	149
	100	225.5	223.9	227.6	216
	150	115.7	85.9	98.6	78.8
	200	130.3	130.4	115.4	99
	250	191	192.3	193.9	185.9
200	50	197.3	224.2	127.3	119
	100	137.4	177.9	183.1	130.4
	150	93.4	72.5	74	63.1
	200	101.4	104.3	91.3	73.4
	250	146.8	176	173.5	139.8

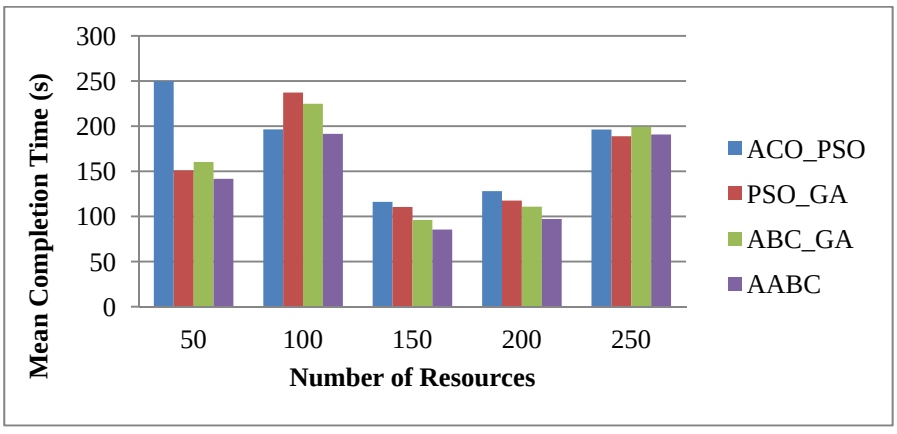

Fig.5. Mean Completion Time for 10 jobs with 50 iterations for ACO_PSO, PSO_GA, ABC_GA and AABC.

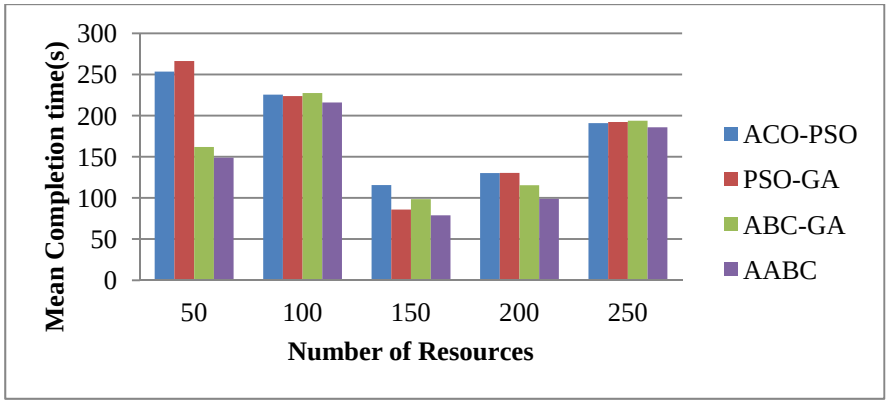

Fig.6. Mean Completion Time for 10 jobs with 100 iterations for ACO_PSO, PSO_GA, ABC_GA and AABC

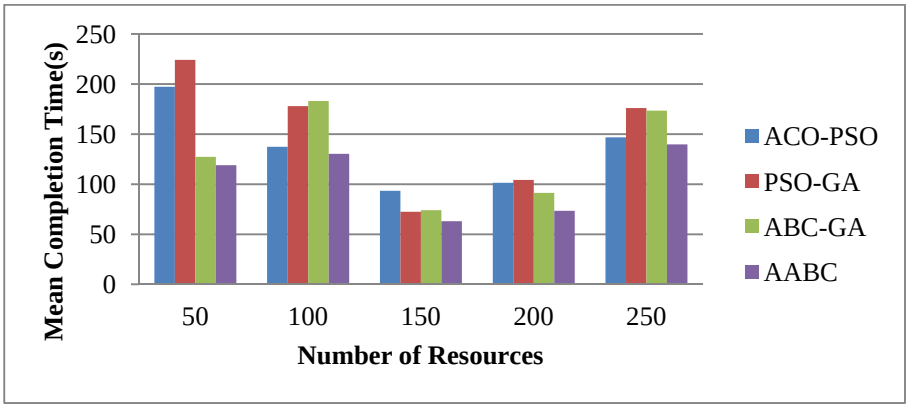

Fig.7. Mean Completion Time for 10 jobs with 200 iterations for ACO_PSO, PSO_GA , ABC_GA and AABC.

VI. CONCLUSION AND FUTURE WORK

A. *Conclusion*

The proposed method AABC has achieved the minimum completion and makes span time. The drawbacks of existing techniques (ACO_PSO, PSO_GA and ABC_GA) were solved by considering few efficient factors in a job scheduling process. In this paper work, an adaptive ABC technique was proposed to allocate the available jobs to the exact resources. Thus, the proposed technique has achieved high performance in allocating the available jobs to the precise resources and also attained a high efficiency. The operation of the proposed job scheduling technique is examined with three hybrid techniques, namely, ABC with GA, PSO with GA and ACO with PSO. The experimental result shows that the proposed job scheduling technique has attained high accuracy and efficiency than the three hybrid techniques viz ACO_PSO, PSO_GA and ABC_GA. In PSO optimization algorithm, a velocity value is utilized for generating the new solutions. Crossover and mutation operators from GA during the replacement process were adopted to examine the utilization process at later stages of the algorithm. These hybrid techniques have performed well with these factors and allocated jobs to the resources for increasing the resource utilization. Hence, the proposed adaptive ABC job scheduling technique is capable of finding the optimal jobs to the resources and also in achieving the minimum completion and makespan time.

B. *Future Work*

In future work the aim is to continue the proposed algorithm in order that it can be given to more pragmatic and integrated manufacturing problems such as dynamic arrivals, machine breakdown, and other elements that affect job status over time. Secondly, an attempt to determine the parameter setting optimally in a dynamic environment, while the algorithm is running may also be undertaken. Finally, the proposed algorithm can be used to solve optimization problems or industrial problems.

References

[1] Rajarshi Mukherjee, Debkalpa Goswami, and Shankar Chakraborty," Parametric Optimization of Nd:YAG Laser Beam Machining Process Using Artificial Bee Colony Algorithm", Journal of Industrial Engineering,Volume 2013 (2013), Article ID 570250, 15 pages.

[2] D.Y. Sha, Hsing-Hung Lin ," A Multi-Objective PSO For Job-Shop Scheduling Problems. Expert Systems With Applications", Volume 37, Issue 2, March 2010, Pages 1065–1070.

[3] Rubiyah Yusof,Marzuki Khalid,Gan Teck Hui,Syafawati Md Yusof,Mohd Fauzi Othman," Solving Job Shop Scheduling Problem Using A Hybrid Parallel Micro Genetic Algorithm". Applied Soft Computing Volume 11, Issue 8, December 2011, Pages 5782–5792.

[4] Sureshkumar, Saravanan.G,S.Thiruvenkadam ," Optimizing Makespan in JSSP Using Unordered Subsequence Exchange Crossover In GA", IOSR Journal of Computer Engineering (IOSR-JCE)e-ISSN: 2278-0661, p- ISSN: 2278-8727Volume 8, Issue 5 (Jan. - Feb. 2013), PP 41-46.

[5] Bin Cai, Shilong Wang;Haibo H,"Hybrid Artificial Immune System for Job Shop Scheduling Problem", World Academy of Science, Engineering and Technology, Vol. 59, No. 18, pp. 81-86, 2011.

[6] Mohammad Akhshabi, Mostafa Akhshabi and Javad Khalatbari," Parallel Genetic Algorithm To Solving Job Shop Scheduling Problem", Journal of Basic Applied Sciences Research, Vol. 1, No. 10, pp. 1484-1489, 2011.

[7] Manish Gupta, Govind Sharma," An Efficient Modified Artificial Bee Colony Algorithm for Job Scheduling Problem", International Journal of Soft Computing and Engineering (IJSCE), Vol. 1, No. 6, pp. 291-296, January 2012.

[8] Ling wang,Gang Zhou,Ye xu;Shengyao Wang,Min Liu,"An Effective Artificial Bee Colony Algorithm for The Flexible Job-Shop Scheduling Problem", Int J Adv Manuf Techno (2012) 60:303-315.

[9] Jun-qing Li, Quan-ke Pan, Sheng-xian Xie, Song Wang,"A hybrid Artificial Bee Colony Algorithm for Flexible Job Shop Scheduling Problems", Int.J.of computers, communication and control, ISSN 1841-9836, Vol.VI (2011) No.2 (june).pp.286-296.

[10] Hazem Ahmed,Janice Glasgow,"Swarm Intelligence: Concepts, Models and Applications", School of Computing,Queen's University, Kingston, Ontario, Canada K7L3N6, February 2012.

[11] C. P. Lim, L. C. Jain, S. Dehuri ,"Innovations in Swarm Intelligence. Studies in Computational Intelligence", Vol. 248, Springer, 2009.

[12] Dervis Karaboga , Bahriye Akay, "A Comparative Study of Artificial Bee Colony Algorithm", Applied Mathematics and Computation - 214 (2009) 108–132.

[13] Li-Ning Xing,Ying-Wu Chen A," Knowledge Based Ant Colony Optimization for Flexible Job Shop Scheduling Problems", Applied Soft Computing, Volume 10 issue 3,june 2010,Pages 888-896.

[14] X. Li, M. Yin," A Discrete Artificial Bee Colony Algorithm With Composite Mutation Strategies for Permutation Flow Shop Scheduling Problem", Scientia Iranica Volume 19, Issue 6, December 2012, Pages 1921–1935.

[15] B.S Girish, N.Jawahar," A Particle Swarm Optimization Algorithm for Flexible Job Shop Scheduling Problem", 5th annual IEEE Conference on Automation science and Engineering Bangalore, India august-2009, Page(s):298 - 303 ,E-ISBN :978-1-4244-4579-0.

[16] Przemyslaw Korytkowski,Szymon Rymaszewski,Tomasz Wisniewski," Ant Colony Optimization for Job Scheduling Using Multi-Attribute Dispatching Rules", Int J Adv Manuf Technol (2013) 67:231-241.

[17] V.Selvi, R.Umarani," The Proposal of an adaptive Hybrid Technique for Job Scheduling Problem", European Journal of Scientific Research, ISSN 1450-216X vol. 93 No 1 December, 2012, pp.45-53.