SOLVING FLOW SHOP SCHEDULING PROBLEM WITH SEQUENCE DEPENDENT SETUP TIME USING HYBRID ARTIFICIAL BEE COLONY ALGORITHM

Jithin K Jayasingh[1], Rajesh Vanchipura[2]
[1]M Tech, Production Engineering, Dept. of Mechanical Engineering ,
Govt. Engineering College, Trissur,
[2]Assistant Professor, Dept. of Mechanical Engineering , Govt. Engineering College, Trissur
Email:jithinjayasinghk@gmail.com[1], rajeshvanchipura@gmail.com[2]

Abstract

This paper presents the development of a hybrid discrete artificial bee colony (HDABC) algorithm for solving flow shop scheduling problem with sequence dependent setup time. The objective is to find a schedule which minimizes the make span. In the HDABC, each solution to the problem is called a food source and represented by a discrete job permutation. The algorithm has two phases namely, employed bee phase and onlooker bee phase. Iterated greedy algorithms comprised of local search procedures based on insertion and swap neighborhood structures are used in both employed bee phase and onlooker bee phase of the algorithm. One of the seed sequence used in the algorithm is developed using well-known NEHRB heuristic and all other solutions are randomly generated. HDABC algorithm is tested on benchmark problems, and finally it is compared with an existing algorithm to evaluate the performance.

Index Terms: flow shop scheduling, sequence dependent, artificial bee colony, tournament selection, local search.

I. INTRODUCTION

As a typical manufacturing and scheduling problem with strong industrial background, flow shop scheduling has appealed wide attention in engineering field. It deals with the allocation of resources to tasks over given time periods and its goal is to optimize one or more objectives. In the present case resources are machines and operations are tasks. Flow shop scheduling problem consists in scheduling some independent jobs on the some consecutive machines. If the jobs have same sequence on all machines, the problem is called permutation flow shop scheduling problem and if the processing sequence on each machine can be altered, the problem is called non-permutation flow shop.

In Flow shop scheduling, there are *n* jobs to be processed on a set of *m* machines where determining the job order is the solution to the problem. The order of the machines do not change. When the setup time is added to it, the complexity of the problem becomes NP complete in nature [1]. For example, a shipyard can be thought of as a collection of several flow-shops. In particular, almost all theparts, i.e. the big metal blocks, go through a panel shop in a ship yard where they are cut or welded together. Panel shops are typically treated as flow shops. The flow-shop problem is easy to describe and formulate, yet computationally it is rather challenging. Therefore, this problem has inspired the development of a number of solution procedures.

Reducing the make span is the main objective that is tried to meet. Since there can be no absolute solution giving method is impossible to formulate for these problems, heuristic methods are widely used. There are constructive and improvement heuristics tried. Constructive heuristic gives a single solution all the time while improvement heuristics give different solutions every time. The advantage is that their algorithm can be easily changed to meet some specific requirements in case they arise. Both have their own merits and demerits. The objective in flow shop scheduling problems is to find a sequence for processing the jobs on the machines so that a given criterion is optimized. This yields a total of $n!$ possible orderings of the operations on each machine and a total of $(n!)^m$ possible processing sequences. In flow-shop scheduling research usually only so called permutation sequences are considered, where the processing order of operations is the same for all machines. We restrict ourselves to the permutation version of the flow shop problem where job passing is not allowed from machine to machine, i.e., the permutation of jobs cannot change from one machine to the next. This results in a smaller solution space of $n!$.

A special case of flow shop scheduling problem is the flow shop scheduling with sequence dependent setup time (SDST), which is selected for present study with minimization of make span as objective (C_{max}).It can be assume that setup time is negligible or part of the job processing time, this assumption simplifies the analysis and/or reflects certain applications, it adversely affects the solution quality of many applications of scheduling that require an explicit treatment of setup times. Hence Flow shop problem with SDST is more realistic in nature than general flow shop scheduling because it consider the setup time between jobs, which is an important parameter to be considered in today's industries. Flow shop with sequence dependent set up time can be represented as, $F/S_{ijk},P_{rmu}/C_{max}$, where the first field indicate that the problem is m machine flow shop problem, and the first part in the second field indicate the SDST and the second part indicate that the permutation is maintained. Finally the third field indicate that the objective function is the completion time. To deal with this problem we have used artificial bee colony algorithm which is a swarm based algorithm, hybridized with local search in an effective manner to yield good result.

There are hundreds of papers on regular flow shop problem meanwhile literature on SDST counterpart is limited. Through this paper we are trying to develop an effective meta-heuristic based on swarm intelligence for flow shop problem with SDSTs. Swarm intelligence has become a research interest to many research scientists of related fields in recent years. The term swarm is used in a general manner to refer to any restrained collection of interacting agents or individuals. The classical example of a swarm is bees swarming around their hive; nevertheless the metaphor can easily be extended to other systems with a similar architecture. An ant colony can be thought of as a swarm whose individual agents are ants. Similarly a flock of birds is a swarm of birds. In this paper our algorithm is based on intelligent foraging behavior of honey bees.

II. LITERATURE REVIEW

Problem of permutation flow shop has got a wide attention during the last few decades and a lot of literatures have been published in this area. But they failed to be a true representative of actual industrial environment as they don't consider setup time between the jobs separately, hence permutation flow shop problems in which setup time between jobs are separately considered is on spotlight during past few years. The present research addresses a specific practical variation of flow shop scheduling problem where the setup time is sequence dependent.

Gupta [1] shows the NP-completeness of flow shop problem with sequence dependent setup time. Hence it is unlikely that there exist a polynomial to solve the problem. The complexity of the problem makes meta-heuristic as the suitable choice for SDST flow shop problems. In comparison to heuristic methods, meta-heuristic methods always obtain better results as they comprised of many different kinds of algorithmic components. A lot of researches are done on regular flow shop problem and

SDST flow shops both of them are considered separately here.

A. Works done on regular flow shop problem

Rajendran and Ziegler [2] applied two ant colony algorithm for permutation flow shop scheduling problem with the objective of minimizing the make span followed by the consideration of minimization of total flow time of jobs. Haq et al. [3] used feed-forward back-propagation artificial neural network (ANN) to solve the problem. Nawaz et al. [4] developed NEH heuristic for the permutation flow shop scheduling problem with the make span minimization criterion. NEH is an effective heuristic for solving the permutation flow shop problem with the objective of make span. It includes two phases: generate an initial sequence and then construct a solution. Dong et al. [5] incorporates an improved NEH based heuristic for solving the problem. Here the initial sequence which is generated by combining the average processing time of jobs and their standard deviations shows better performance. Lin and Ying [6] developed a multi-start simulated annealing (MSA) heuristic, which adopts a multi-start hill climbing strategy in the simulated annealing heuristic to obtain near optimal solutions.

B. Works done on flow shop scheduling problems with SDST.

As compared with the regular flow shop scheduling problem, the research work carried out on SDST flow shop is less. Although a number of articles on SDST flow shops are published in last five years. Ruiz et al. [7] present two advanced genetic algorithms as well as several adaptations of existing advanced meta-heuristics that have shown superior performance when applied to regular flow shops. Gajpal et al. [8] have developed a new ant colony algorithm in their paper to solve the flow shop scheduling problem with the consideration of sequence dependent setup times of jobs. The objective is to minimize the make span. Artificial ants are used to construct solutions for flow shop scheduling problems, and the solutions are subsequently improved by a local search procedure. Rajendran and Ziegler [9] in their paper "Scheduling to minimize the sum of weighted flow-time and weighted tardiness of jobs in a flow shop with sequence-dependent setup times", two heuristic preference relations are used to construct a good heuristic permutation sequence of jobs. Thereafter, an improvement scheme is implemented, once and twice, on the heuristic sequence to enhance the quality of the solution. Salmasi et al. [10] have developed a mathematical programming model for minimizing total flow time of the flow shop sequence dependent group scheduling problem. A tabu search algorithm as well as a hybrid ant colony optimization (HACO) algorithm have been developed to heuristically solve the problem along with a lower bounding method based on the Branch-and-Price algorithm is also developed to evaluate the quality of the meta-heuristic algorithms.

III. PROBLEM FORMULATION

The present study deals with permutation flow shop and considers setup time between the jobs. The problem involves scheduling *n* jobs on m machines, where each job passes through a machine once and only once. The setup time of jobs are sequence dependent which means the setup time of a job *j*, on a machine is dependent on job *k* processed just prior to job *j* on the machine. The order of the machines do not change. It is the order of the jobs which changes and needs to be optimized.

While formulating the problem, the following assumptions are used.

- Any job can be processed at any position in the schedule.
- Only one job is processed in one machine at a time.
- No job can be taken out before it is completed.
- There is a setup time for every job which is dependent of the previous job.
- All machines are available at all time.

The notations used are

- m Total number of machines
- n The total number of jobs
- i Index of the machine
- j Index of the job
- P_{ij} Processing time of job j on machine i
- S_{ijk} Setup time on machine i when job k is preceded by job j

- Q(σ, i) Completion time for partial sequence σ on machine i
- Q(σj, i) Completion time of job j on machine i, when job j is appended on the partial sequence

The total number of jobs considered is n. However, a dummy job is added ahead of every sequence. A dummy job consumes no processing time. It is included in the sequence to consider the setup time required for the first job in the sequence. For calculating the start and completion times of jobs on machines in the permutation flow shop, recursive equations are used as follows.

The completion time of σj on machine i is determined using the following recursive equation,

$$q(\sigma j,i) = \max(q(\sigma,i)+ S_{ijk}, q(\sigma j,i-1)) + Pij \quad (1)$$

where q (Φ, i) = 0 and q (σ, 0) = 0, for all σ and i, with Φ denoting a null schedule. It is assumed that S_{ijk} exists for all jobs where j = Φ for all machines. It is also assumed that setup of a machine can be done without the job being available at the machine.

The flow time of job j, C_j, is given by

$$C_j = q(\sigma j, m) \quad (2)$$

When all the jobs are scheduled, the make span M is obtained as follows.

$$M= \max\{C_j,j= 1,2,3,\ldots\ldots,n\} \quad (3)$$

IV. METHODOLOGY

Swarm intelligence has become a research interest to many research scientists of related fields in recent years. Swarm intelligence can be defined as "any attempt to design algorithms or distributed problem-solving devices inspired by the collective behavior of social insect colonies and other animal societies"[11]. The classical example of a swarm is bees swarming around their hive; nevertheless the metaphor can easily be extended to other systems with a similar architecture. In the present work foraging behavior of honey bees are used to design an algorithm to solve flow shop scheduling problem considering sequence dependent setup time.

A. Artificial bee colony algorithm

The minimal model of forage selection that lead to the emergence of collective intelligence of honey bee swarms consists of three essential components: food sources, employed foragers and unemployed foragers.

i) Food Sources: The value of a food source depends on many factors such as its proximity to the nest, its richness or concentration of its energy, and the ease of extracting this energy.

ii) Employed foragers: They are associated with a particular food source which they are currently exploiting or are "employed" at. They carry with them information about this particular source, its distance and direction from the nest, the profitability of the source and share this information with a certain probability.

iii) Unemployed foragers: They are continually at look out for a food source to exploit. There are two types of unemployed foragers: scouts, searching the environment surrounding the nest for new food sources and onlookers waiting in the nest and establishing a food source through the information shared by employed foragers. The mean number of scouts averaged over conditions is about 5-10%.

The exchange of information among bees is the most important occurrence in the formation of collective knowledge. While examining the entire hive, it is possible to distinguish some parts that commonly exist in all hives. The most important part of the hive with respect to exchanging information is the dancing area. Communication among bees related to the quality of food sources occurs in the dancing area. The related dance is called waggle dance. Since information about all the current rich sources is available to an onlooker on the dance floor, she probably could watch numerous dances and choose to employ herself at the most profitable source. There is a greater probability of onlookers choosing more profitable sources since more information is circulating about the more profitable sources. Employed foragers share their information with a probability, which is proportional to the profitability of the food source, and the sharing of this information through waggle dancing is longer in duration. Hence, the recruitment is proportional to profitability of a food source.

In order to understand the basic behavior characteristics of foragers better, let us examine

Figure 1. Assume that there are two discovered food sources: A and B. At the very beginning, a potential forager will start as unemployed forager. That bee will have no knowledge about the food sources around the nest. There are two possible options for such a bee:

i) It can be a scout and starts searching around the nest spontaneously for a food due to some internal motivation or possible external clue (S on Figure 1).

ii) It can be a recruit after watching the waggle dances and starts searching for a food source (R on Figure 1).

After locating the food source, the bee utilizes its own capability to memorize the location and then immediately starts exploiting it. Hence, the bee will become an "employed forager". The foraging bee takes a load of nectar from the source and returns to the hive and unloads the nectar to a food store. After unloading the food, the bee has the following three options:

i) It becomes an uncommitted follower after abandoning the food source (UF).

ii) It dances and then recruits nest mates before returning to the same food source (EF1)

iii) It continues to forage at the food source without recruiting other bees (EF2).

Davis karaboga inspired by the intelligent foraging behavior of the honey bee swarm developed artificial bee colony algorithm in 2005. Artificial bee colony algorithm also consist of three basic elements: food sources, employed foragers and unemployed foragers. But this cannot directly applied on the present problem since it is for continuous optimization problem and not for discrete problem like flow shop scheduling.

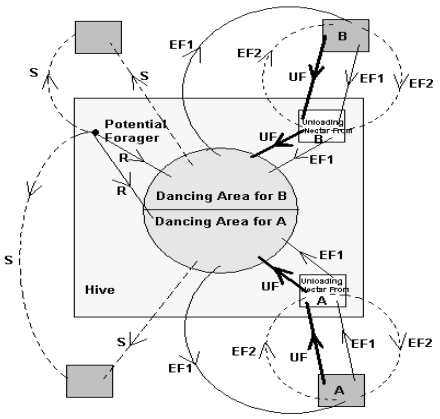


Fig. 1 behavior of bees in nature.

B. Hybrid artificial bee colony algorithm

Hybrid artificial bee colony algorithm is basically an iterative process with all the three phases of algorithm repeats until the stopping criteria is reached.

1. Initialize the solutions
2. Send the employed bees to the solutions
3. Find the neighborhood of the solution and replace the parent solution if the neighbor is better than parent, apply local search if the probability value is higher than 0.5.
4. Send the onlooker bees to the discovered solutions based on the probability value.
5. Find the neighborhood of each solutions visited by onlooker and replace the parent solution with the neighbor and sequentially apply local search if the probability value is higher than 0.5.
6. Replace the worst solution in the sequence with the solution found by tournament selection.
7. Go to step 2 if the termination criteria is not reached.

1. Initialization

In hybrid artificial bee colony algorithm for flow shop scheduling, the solutions are a sequence of jobs and can be represented as $j = \{j1,j2,j3,\ldots\ldots\ldots,j_n\}$. Since hybrid artificial bee colony algorithm is an improvement heuristic the searching for a near optimum solution starts from a set of initial solution. One of the solution in the initial population is generated using well known NEHRB heuristic which is an improvement heuristic and all other solutions are generated randomly.

NEH Algorithm consist of 4 steps

Step 1: Order the jobs by non-increasing sums of processing times on the machines;

Step 2: Take the first two jobs and schedule them in order to minimize the partial make span as if there were only these two jobs

Step 3: For k= 3 to n do Step 4

Step 4: Insert the k^{th} job at the place, which minimizes the partial make span among the k possible sequences.

2. Employed bee phase

Number of employed bees are set to half the population size which is equal to 40. Each Employed bees generate a neighbor to its corresponding solution in search for improved solution. For this purpose we use six strategies

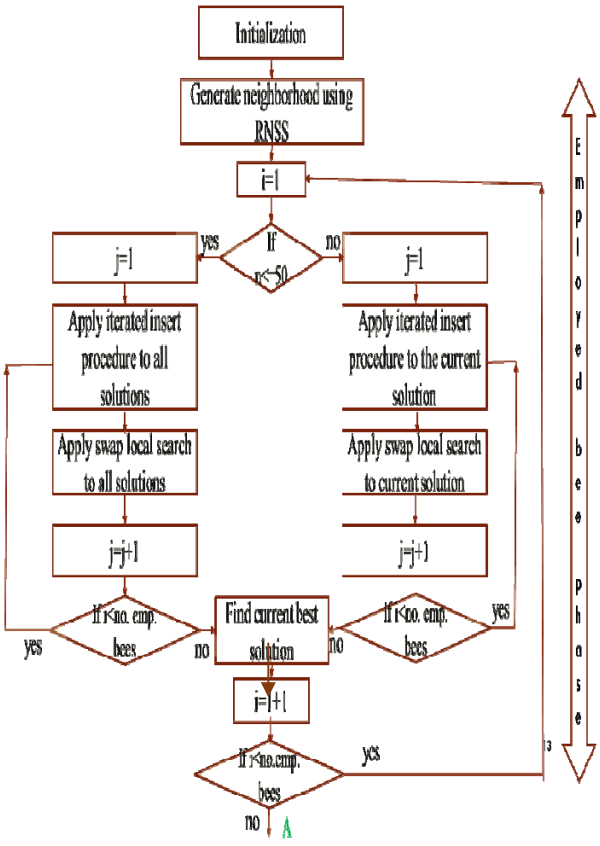
and apply one of them randomly to each sub population which make HABC algorithm a multi populated algorithm. We use insert, swap and destruction construction procedure to generate a neighbor. Insert procedure basically remove a job from its current position and insert it to another position. Whereas in swap operation we interchange the position of two jobs. Construct destruct process which can be thought of as a part of NEH heuristic which consist of two phases in the first phase a set of jobs are removed from the sequence which is called destruction as we split the original sequence into two parts destructed set and constructed set. The second phase called as construction consist of inserting all the jobs from destructed set to the partial sequence one at a time to all the possible positions and select a position which minimize the make span. By suitably selecting the values for perturbation strength p of insert and swap operation and destruction size of construct destruct procedure we obtain the six strategies to generate the neighbor for the solutions. The perturbation strength selected for insert and swap operation are 1 and 2 while the destruction size selected for construct destruct procedures are 2 and 4. After the neighborhood is generated using repeated neighborhood search strategy a local search in which insertion local search is carried on each solution in a well-organized manner if the probability value is higher than 0.5. The pseudo code for insert local search and swap local search are shown in figure 4 and figure 5 respectively.

3. Onlooker bee phase

The number of onlooker bees are same as that of employed bees. Onlookers select a food source from those found out by employed bees. This selection is done in two ways, half of the onlookers select the food source depending upon their profitability which is achieved by means of tournament selection and the other half select those food source corresponding to their employed bees. In tournament selection some individuals are selected from the solution set and the solution having higher probability is get selected. Hence solution having larger probability values will survive. The motivation behind using tournament selection in onlooker bee is that we could intensify the search near promising solution by sending more and more onlooker bees to that solution. Also it is onlooker bee phase which brings the balance between intensification and diversification, a must need quality of a good algorithm. After selecting food sources next we apply repeated neighborhood strategy to generate a neighboring solution. If the neighbor is better than the current solution then it will become the current solution. A local search procedure in which insertion local search and swap local search applied sequentially is then carried out on the solutions in case of higher problem sizes in which the number of jobs are more than 50,the same in a repeating manner if number of jobs are higher than 50.

4. Scout bee phase

In the basic artificial bee colony algorithm scout bee search for a food source around the hive without having any information about any food source. But here the solutions in the populations are relatively better, hence one of the food source from the population is selected for scout bee. The scout bee using tournament selection of tour size two, select two food sources from the population and replaces the worst solution in the population with this solution.



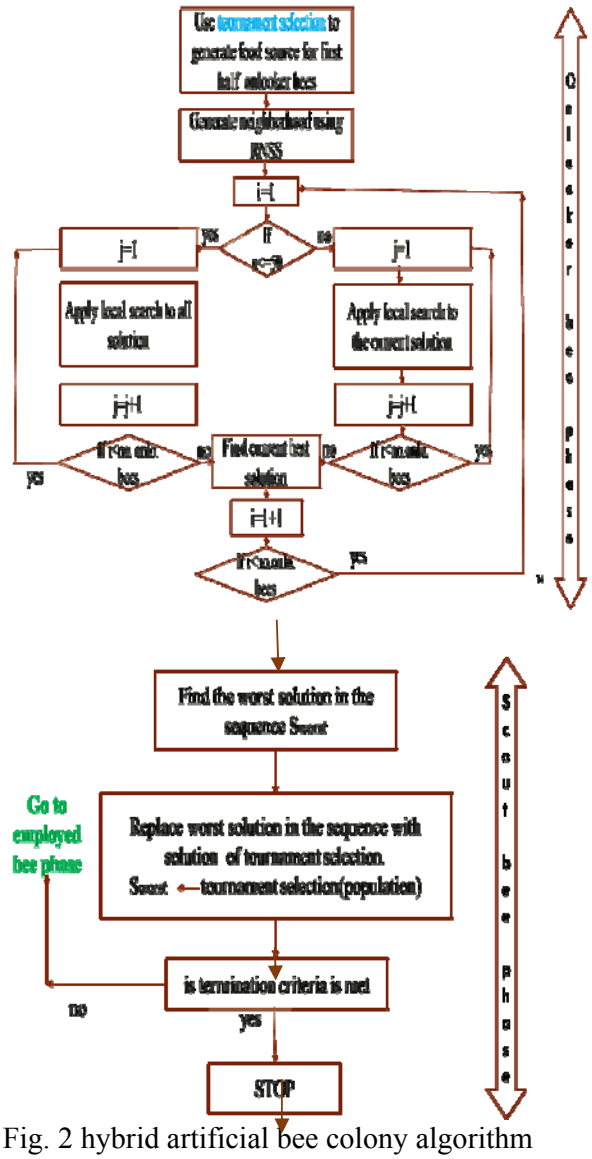


Fig. 2 hybrid artificial bee colony algorithm

V. RESULTS

Hybrid artificial bee colony algorithm is developed and tested on 30 SDST benchmark problem instances and compared the performance with genetic algorithm. The algorithm was coded in MATLAB and run in a PC of core i3 processer and 4 GB RAM.

Figures 3, 4 and 5 show the makespan results obtained using the developed heuristic. It shows that the HDABC is better all the 30 problem instances considered in the present study.

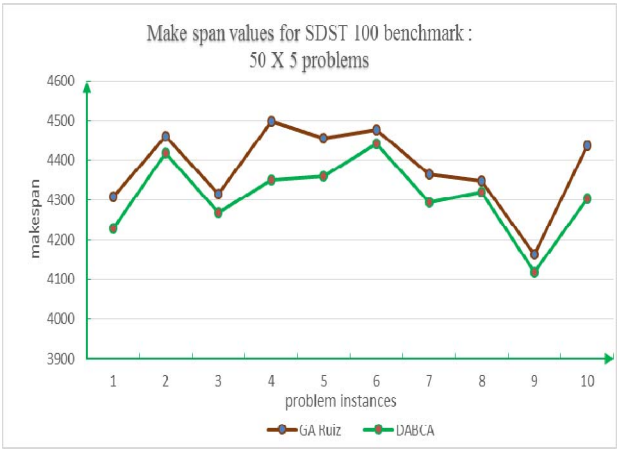


Fig.3 make span value of ten 50*5 problem instances.

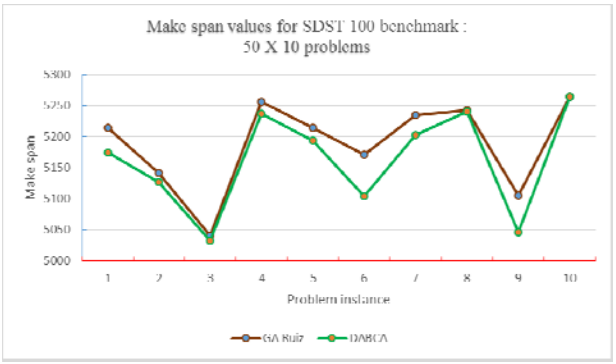


Fig.4 make span value of ten 50*10 problem instances

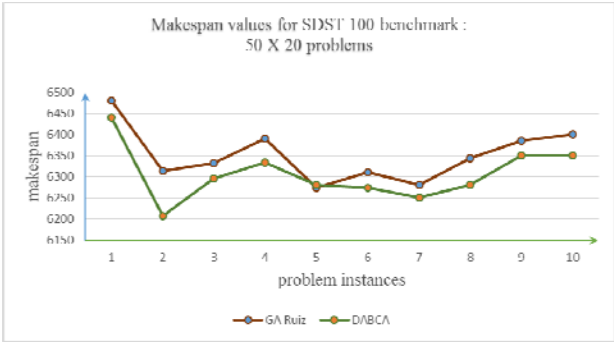


Fig.5 make span value of ten 50*20 problem instances

VI. CONCLUSION

In this paper presents the development of a hybrid artificial bee colony algorithm for solving flow shop scheduling problem with sequence dependent setup time. The proposed algorithm is used to solve the SDST flow shop scheduling problem. 30 problem instances are taken from the SDST flow shop benchmark problem for carrying out the experimentation. In the performance analysis the makespan value obtained using the HDABC algorithm is compared that obtained using an existing hybrid

genetic algorithm. The proposed algorithm is found to outperform genetic algorithm almost in all the problem instances. For future research hybrid discrete artificial bee colony (HDABC) algorithm can be extended to solve similar category of problems such as multi criteria flow shop scheduling problem with sequence dependent setup time.

VII. REFERENCES

[1] Jatinder N. D. Gupta, flowshop schedules with sequence dependent setup times, Journal of the Operations Research Society of Japan Vol. 29, No. 3, September,1986.

[2] Chandrasekharan Rajendran ,Hans Ziegler, Ant-colony algorithms for permutation flowshop scheduling to minimize makespan/total flowtime of jobs, European Journal of Operational Research 155 (2004) 426–438.

[3] A. Noorul Haq, Radha Ramanan T, Sarang Kulkarni, R. Sridharan, A hybrid neural network–genetic algorithm approach for permutation flow shop scheduling, International Journal of Production Research 48(14):4217-4231 July 2010.

[4] Nawaz M, Enscore Jr. EE, Ham I., A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem, OMEGA, The International Journal of Management Science, **11**, *1*, pp. 91–95, 1983.

[5] X.Y. Dong, H.K. Huang, P. Chen, An improved NEH-based heuristic for the per-mutation flowshop problem, Comput. Oper. Res. 35 (2008) 3962–3968**.**

[6] S.W. Lin, K.C. Ying, Minimizing makespan and total flow time in permutation flow shops by a bi-objective multi-start simulated-annealing algorithm, Com-put. Oper. Res. (2011).

[7] Ruben Ruiz , Concepcion Maroto, Javier Alcaraz, Solving the flow shop scheduling problem with sequence dependent setup times using advanced metaheuristics, European Journal of Operational Research 165 (2005) 34–54.

[8] Yuvraj Gajpal, Chandrasekharan Rajendran, Hans Ziegler, An ant colony algorithm for scheduling in flow shops with sequence dependent setup times of jobs, The International Journal of Advanced Manufacturing Technology September 2006, Volume 30, Issue 5.

[9] Chandrasekharan Rajendran and Hans Ziegler, Scheduling to minimize the sum of weighted flowtime and weighted tardiness of jobs in a flow shop with sequence-dependent setup times, European Journal of Operational Research 149(3):513-522.February 2003.

[10] Nasser Salmasi , Rasaratnam Logendranb, Mohammad Reza Skandari, Total flow time minimization in a flow shop sequence-dependent group scheduling problem, Computers & Operations Research 37 (2010) 199 – 212.