VOL. 8, NO. 2, FEBRUARY 2013 ISSN 1819-6608
ARPN Journal of Engineering and Applied Sciences


www.arpnjournals.com

ENHANCED BACTERIAL FORAGING ALGORITHM FOR PERMUTATION FLOW SHOP SCHEDULING PROBLEMS

Shivakumar B. L.[1] and Amudha T. [2]
[1]Department of Computer Applications, Sri Ramakrishna Engineering College, Coimbatore, India
[2]Department of Computer Applications, Bharathiar University, Coimbatore, India
E-Mail: amudha.swamynathan@gmail.com

ABSTRACT

Biologically Inspired algorithms are a kind of algorithms that imitate the problem solving behavior from biological species. Bio-inspired computing is a subset of Nature-inspired computing that focus on social behavior and emergence of biological species. Bacterial Foraging algorithm is a relatively new biologically inspired optimization technique based on the foraging behaviour of E. coli bacteria. This paper deals with one of the significant types of scheduling problems, the permutation flow shop scheduling problem. The competence of bacterial problem solving and a proposed hybrid bacterial swarming technique were analyzed by applying them to benchmark problems of permutation flow shop. A comparative analysis of the results indicates the improvement in scheduling efficiency in terms of reduced cost through the application of bio-inspired techniques.

Keywords: bio-inspired computing, bacterial foraging algorithm, permutation flow shop, genetic algorithm.

1. INTRODUCTION

Bio-inspired computing has originated by observing the social behavior of the biological species such as ants, wasps, termites, fish schools, and flock of birds and so on. Functioning as a colony with specialized duties, social organisms are found to be more efficient in getting necessary jobs done such as finding food, nest building, self organization, larvae sorting, division of labor and cooperative transportation. Bio-inspired algorithms were experimentally proven to be highly capable when compared with the other contemporary approaches such as simulated annealing, tabu search, genetic algorithms etc., Scheduling and task allocation in a non-deterministic environment are currently carried out under real world conditions by using bio-inspired approaches as well as intentional approaches. Although both the approaches are robust, efficient and competent to each other, bio-inspired approaches seem to be superior due to their decentralized and distributed nature [1]. Bio-inspired approaches perform better even when the knowledge about task and their states are not exactly available. In a social insect colony, the various individuals tend to be specialized in particular tasks. But, this specialization is very much flexible and the individuals do different tasks so as to adjust themselves according to the changing, uncertain, unpredictable conditions [2].

Scheduling problems are one of the most important problems in the field of combinatorial optimization (CO), and find their application in various engineering and manufacturing industries. They are generally considered as NP-hard CO problem [3]. Scheduling can be defined as the problem of deciding which job in which order to be done with which machine, when several jobs are processed with several machines. In general, the scheduling problem is classified into several types, when the problem involves multiple machines. The Permutation Flow Shop Scheduling Problem (PFSSP) is one of the significant problems where a set of jobs flow through multiple stages in the same order. Many noteworthy research efforts are being devoted for sequencing jobs in a flow shop thereby minimizing the make span [4, 5, 6, 7]. The permutation flow shop problem belongs to the category of NP-hard scheduling problems and hence, the search for an optimal solution has gained equally theoretical and practical importance [8].

Permutation flow shop scheduling is a computationally complex production problem where a set of n jobs are to be processed with identical flow pattern on m machines [9]. The processing times are fixed, nonnegative, and may be zero if a job is not processed on some machine. Further assumptions are that each job can be processed on only one machine at a time, the operations are not pre-emptive, the jobs are available for processing at time zero and setup times are sequence independent. Since there is no job passing, the number of possible schedules for n jobs is $n!$. The scheduling performance measure is related to an efficient resource utilization looking for a job sequence that minimizes the makespan; that is the total time to complete the schedule. The objective of this problem is to find a sequence, i.e., a permutation of the numbers 1, 2 …n that minimizes the makespan.

In this paper, bacterial swarming technique is hybridized with Genetic Algorithm (GA) thereby modifying the swarming behavior of the bacteria. Bacterial Swarming Algorithm (BSA) and the proposed hybrid bacterial technique using GA, named as Genetic Bacterial Swarming Algorithm (GBSA) are applied to solve various benchmark instances of PFSSP. PFSSP benchmark instances were taken from E. Taillard. These instances are worth solving as they are considered as the most significant, classical and complex CO problems known to be NP-hard. The effectiveness of hybrid Bacterial swarming methodology, GBSA is analyzed by comparing its performance with Bacterial foraging method. Experimental results on various benchmark

instances have highlighted the capability of the improved bio-inspired method in solving complex scheduling problems and the improvement in optimal solutions when BSA was hybridized with GA.

2. RELATED WORK

Seda Hezer and Yakup Kara [10] have proposed an idea about Solving Vehicle Routing Problem with Simultaneous Delivery and Pick-up using BFO Algorithm. Solving Vehicle Routing Problem with Simultaneous Delivery and Pick-up is a NP-Hard Combinatorial Optimization problem. In this study, a heuristic solution approach based on BFO algorithm was used to minimize the total distance travelled and the results have been compared with the insertion based heuristic algorithm.

W. J. Tang *et al*., [11] have proposed a bacterial foraging algorithm (BFA) aiming for optimization in dynamic environments, called DBFA. A test bed proposed previously was adopted to evaluate the performance of DBFA. The simulation studies offered a range of changes in a dynamic environment. The simulation results showed that DBFA could adapt to various environmental changes which occurred in different probabilities, with both satisfactory accuracy and stability, in comparison with another recent work on bacterial foraging.

Zulkifli Zainal Abidin *et al*., [12], have reviewed many of the recent Animal-Inspired Metaheuristic algorithms such as the Ant Colony Optimization, the Bee Algorithm, the Monkey Search Algorithm and Firefly Algorithm. In their paper, the authors have presented a Fly Optimization Algorithm, a new member in the metaheuristic family. They have demonstrated the intelligence of the fly in finding the optimal path while searching for food and searching for mate. They also have given a brief comparison on the theoretical behaviors among the animal inspired algorithms.

Wei Liu *et al*., [13] have presented a novel optimal scheduling method for Radio Frequency Identification (RFID) network using a Self-adaptive Bacterial Foraging Optimization Algorithm. The method, which was based on the Bacterial Foraging Optimization technique, could adjust the run-length unit parameter dynamically during evolution to balance the exploration/exploitation tradeoff. The simulation results, when compared to GA, PSO and BFO, have shown that this technique has obtained superior solutions than the other methods.

Sambarta Dasgupta *et al*., [14] have introduced a micro-bacterial foraging optimization algorithm, which evolved with a very small population compared to its classical version. In this modified bacterial foraging algorithm, the best bacterium was kept unaltered, whereas the other population members were reinitialized. This new small population μ-BFOA was tested over a number of numerical benchmark problems for high dimensions and was found to outperform the normal bacterial foraging with a larger population as well as with a smaller population.

Dong Hwa Kim *et al*., [15] have proposed a hybrid approach involving genetic algorithms and bacterial foraging algorithms for function optimization problems. They illustrated the proposed method using four test functions and the performance of the algorithm was studied with an emphasis on mutation, crossover, variation of step sizes, chemotactic steps, and the lifetime of the bacteria. The proposed algorithm was then used to tune a PID controller of an automatic voltage regulator. Simulation results clearly illustrated that the proposed approach was quite efficient and could easily be extended for other global optimization problems.

E. Taillard [16] has proposed a paper about Benchmarks' for Basic Scheduling Problems. In this paper he discussed about 260 scheduling problems whose sizes correspond to real dimensions of industrial problems. Ashwani Kumar Dhingra [6] gave a brief explanation about scheduling problems, significance of scheduling, scheduling in a manufacturing system and classification of scheduling problems based on requirement generations. Problems up to 200 jobs and 20 machines for instances developed by Taillard have been solved in this paper and also proposed some metaheuristics that can be tested on other types of problems.

Samiakouki, Mohamed Jemni *et al*., [17] have analyzed the solution methods for Permutation Flow Shop Problem with Makespan Criterion using Grids. The optimization of scheduling problems was usually based on different criteria and one of the most important criteria is the minimization of completion time of the last task on the last machine called makespan. They presented a parallel algorithm for solving the permutation flow shop problem. This was used to minimize the total makespan of the tasks by using Branch and Bound method to find optimal solutions.

Mircea Ancau [9] has proposed two variants for heuristic algorithms; the constructive greedy heuristic algorithm and stochastic greedy heuristic algorithm to solve the classic Permutation Flow shop Scheduling Problems. The proposed algorithms were tested by using four groups of benchmark problems proposed by Taillard, Carlier, Heller's and Reeves, Taillard and Reeves. Both the variants were simple and also offered good quality solutions. Due to its relatively high computational degree, the proposed heuristics require several hours of CPU time for very large size problems, which might be costly. The author recommended these variants mainly for medium size problems.

3. PROPOSED GBSA FOR FLOW SHOP SCHEDULING

In this paper, we propose a hybrid technique where the search space exploration is done by Bacterial Swarming Algorithm and exploitation within the explored space is done by GA which was aimed at providing improved search performance in scheduling environments. In our proposed GBSA, the bacterial swarming technique is used as the constructive heuristics for generation of the initial population and specific recombination methodology

is devised as per the constraints of the scheduling problem. Then GA is employed as the improvement heuristic to be applied to the population and swarming surface built by the BSA. BSA identifies a potential swarming surface and identifies an initial locally optimal solution. After the creation of the initial population, the standard procedures of GA, selection, crossover and mutation take place. The significant aspect of GBSA is that it manages with the optimized solutions as inputs to the selection phase. It was of general opinion from the researchers that an improvement choice over the crossed over solutions will be effective to achieve competitive performance. The mutation is then performed on the locally optimized offspring.

Pseudocode of the proposed hybrid GBSA

0 procedure GBSAMetaheuristic ( )
1 for Elimination-dispersal loop do
2 for Reproduction loop do
3 for Chemotaxis loop do
4 for Bacterium i do
5 Tumble: Generate a random vector φ □ R^D in rand direction
6 Compute new move strategy and move in the rand direction
7 Compute additional cost function for swarming
8 Initialize Swim length
9 while Swim length< No of steps do
10 perform a swim
11 if the new direction is promising then
12 compute swarming positions and initialize optimized population
13 for No. of decision variables do
14 Evaluate the individuals
15 Perform selection
16 Perform crossover based on probability condition
17 Perform a swim or tumble of bacteria over the resultant of crossover
18 Perform mutation, if required based on probability condition
19 Update swarming surface based on new search space
20 Compute the next bacteria move
21 end
22 else
23 Let Swim length = no. of steps
24 end
25 end
26 end
27 end
28 Sort bacteria as per ascending cost
29 Split the group ,bacteria with highest J value die and bacteria with best value split
30 Update value of objective function and swarm nutrient cost accordingly
31 end
32 Eliminate and disperse the bacteria to random locations with probability p_{ed}
33 Update corresponding objective function and swarm nutrient locations
34 End

The solution obtained after mutation is again transformed into an optimized solution to keep the local optimality of the population. At this stage, mutation takes responsibility to minimize the possibility of return into previous local optima. The population replacement and swarming over a new and diverse region in the search space are specific to the bacterial strategy.

Bacterial swarming strategy guarantees a satisfactory degree of the diversity of the population, thereby trying to keep away from the drawbacks of GA such as premature convergence. The hybridization of BSA and GA has shown a remarkable improvement over the solution generation. This is mainly due to the fact that the population of local optima is maintained. Search space diversification, solution reconstruction and the periodic local improvement are the primary phases of this hybrid technique. High quality results could be obtained by repeating these phases several times. At every chemotactic step, these primary phases are performed. Then a new population is created by reproduction of eligible bacteria in the population. If the level of the diversity of certain individuals is found to be lesser than a certain threshold, they are eliminated and the fittest individuals are dispersed once again to continue with the next run. The following Table-1 depicts the parameter settings for our proposed GBSA.

4. EXPERIMENTAL RESULTS

This section analyzes the results of the implementation of Bacterial foraging technique and our proposed hybrid metaheuristics, GBSA in solving benchmark instances of Permutation Flow Shop Scheduling Problems. The results have shown that our proposed method is highly successful in solving the benchmark problems, especially the large benchmark instances which have been established as the most difficult problem classes. In this research paper, the benchmark problem instances for Permutation Flow Shop Scheduling Problems generated by E. Taillard [16] are used. E. Taillard has generated problem instances with 20 jobs, 50 jobs, 100 jobs, 200 jobs and 500 jobs. A total of 90 problem instances with 20 jobs, 50 jobs and 100 jobs are used in this work for testing our bio-inspired techniques. Table-2, Table-3 and Table-4 shows the makespan comparison of PFSSP benchmark instances of size 20, size 50 and size 100, respectively.

The problem instances proposed by E. Taillard were highly complex and challenging. Bacterial Swarming Algorithm could achieve near-optimal makespan to the known lower bound only in very few cases whereas our proposed techniques, GBSA could attain highly noticeable improvement in makespan. The relative gap analysis with respect to the known lower bound and upper bound was done for the resultant makespan obtained by BSA and proposed GBSA techniques. The minimum makespan deviation, maximum makespan deviation and mean makespan deviation of all the three techniques were identified with respect to the known lower bound and upper bound reported in the literature.

By comparing the overall makespan deviation of the methods for all the PFSSP instances considered in this research work, it is quite obvious that GBSA is highly competent and have achieved better makespan than the

Bacterial Swarming Algorithmic technique. GBSA has achieved nearly 78% of optimal solutions much closer to the lower bound, whereas BSA got only 26% of its solutions closer to the lower bound. The figures, Figure-1, Figure-2 and Figure-3 show the minimum, maximum and mean relative gap to known lower bound.

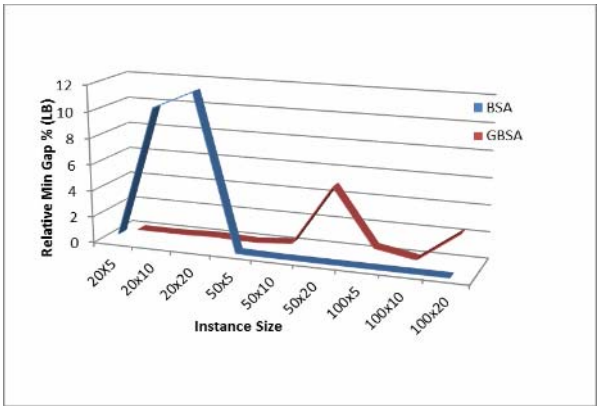

Figure-1. Relative minimum gap with lower bound.

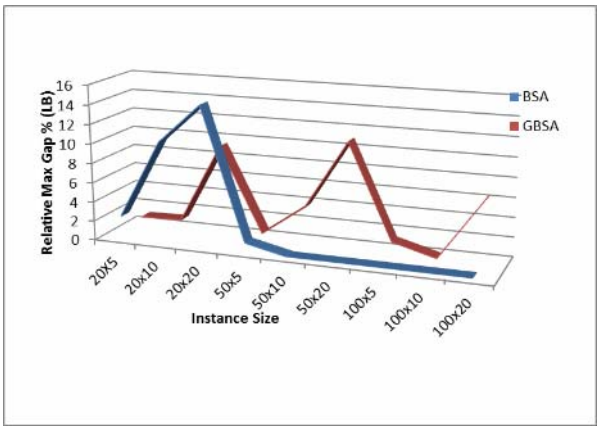

Figure-2. Relative maximum gap with lower bound.

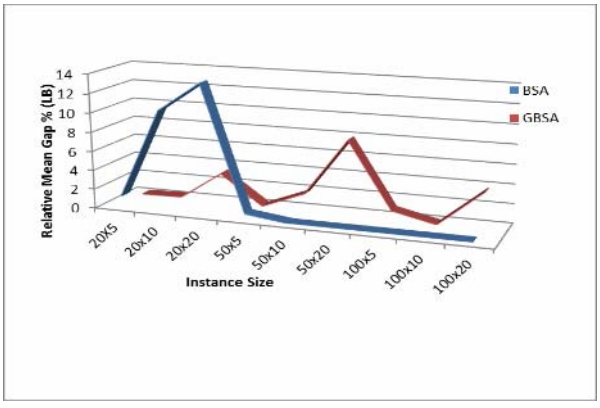

Figure-3. Relative mean gap with lower bound.

Figure-4, Figure-5 and Figure-6 depicts the maximum, minimum and mean relative gap comparison among BSA and GBSA with the known upper bound.

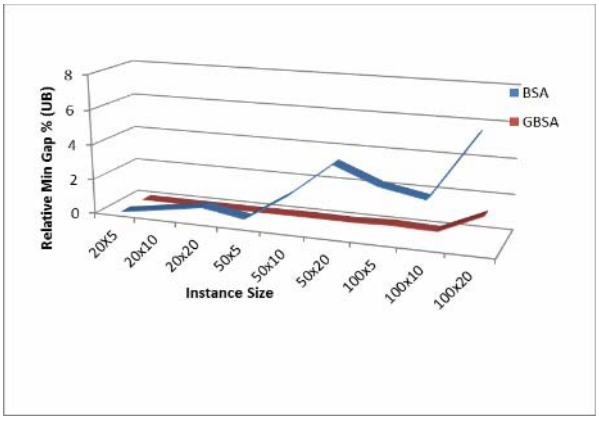

Figure-4. Relative minimum gap with upper bound.

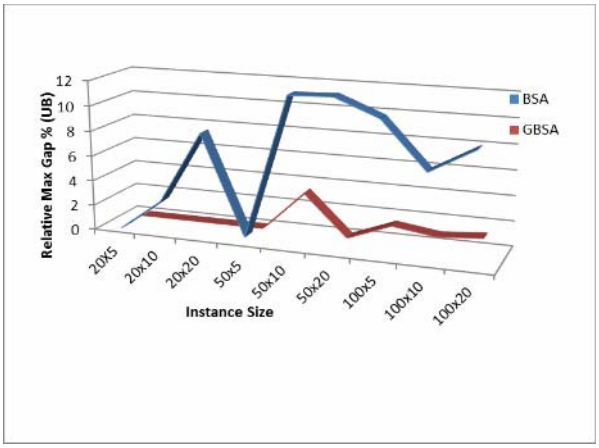

Figure-5. Relative maximum gap with upper bound.

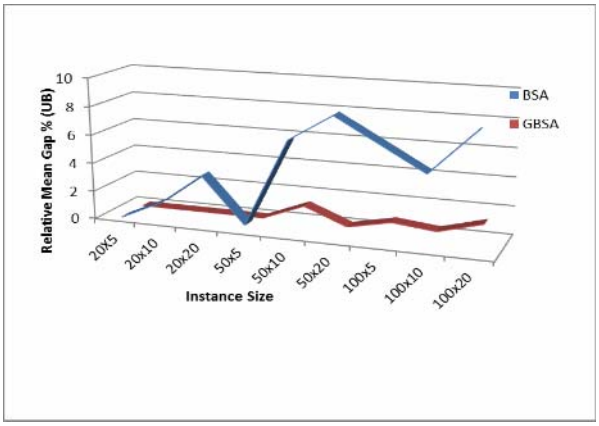

Figure-6. Relative mean gap with upper bound.

Table-1. Parameter settings for proposed GBSA.

Parameter	Values	Description
S	50	No. of bacterium
N_c	30	No. of chemotactic steps
Ns	4	Swim length
N_{re}	4	No. of reproduction steps
N_{ed}	5	No. of elimination dispersal events
$d_{attract}$	0.1	Depth of attractant
$w_{attract}$	0.2	Width of attractant
$h_{repellant}$	0.1	Height of repellent
$w_{repellant}$	10	Width of repellent
P_{ed}	0.05	Probability of elimination-dispersal
Crossover ratio	0.8	Probability of crossover
Mutation ratio	0.04	Probability of mutation
T size	5	Tournament size
Generations	500 - 3000	Number of generations in GA

Table-2. Makespan comparison of PFSSP benchmark instances (Size 20).

Instance Name	Instance size	Makespan		Upper bound	Lower bound
		BSA	GBSA		
Tai_20x5_1	20x5	1245	1232	1278	1232
Tai_20x5_2		1299	1290	1359	1290
Tai_20x5_3		1079	1074	1081	1073
Tai_20x5_4		1278	1268	1293	1268
Tai_20x5_5		1218	1198	1236	1198
Tai_20x5_6		1187	1182	1195	1180
Tai_20x5_7		1234	1227	1239	1226
Tai_20x5_8		1198	1182	1206	1170
Tai_20x5_9		1225	1212	1230	1206
Tai_20x5_10		1098	1085	1108	1082
Tai_20x10_1	20x10	1592	1458	1582	1448
Tai_20x10_2		1689	1481	1659	1479
Tai_20x10_3		1506	1410	1496	1407
Tai_20x10_4		1399	1308	1378	1308
Tai_20x10_5		1456	1326	1419	1325
Tai_20x10_6		1426	1290	1397	1290
Tai_20x10_7		1516	1397	1484	1388
Tai_20x10_8		1522	1379	1538	1363
Tai_20x10_9		1599	1474	1593	1472
Tai_20x10_10		1602	1356	1591	1356
Tai_20x20_1	20x20	2167	2108	2297	1911
Tai_20x20_2		2290	1867	2100	1711
Tai_20x20_3		2378	1858	2326	1844
Tai_20x20_4		2278	1920	2223	1810
Tai_20x20_5		2342	1984	2291	1899
Tai_20x20_6		2376	1922	2226	1875
Tai_20x20_7		2184	1874	2273	1875
Tai_20x20_8		2198	1887	2200	1880
Tai_20x20_9		2302	1842	2237	1840
Tai_20x20_10		2195	1916	2178	1900

Table-3. Makespan comparison of PFSSP benchmark instances (Size 50).

Instance Name	Instance size	Makespan		Upper bound	Lower bound
		BSA	GBSA		
Tai_50x5_1	50x5	2734	2712	2724	2712
Tai_50x5_2		2820	2809	2836	2808
Tai_50x5_3		2618	2603	2621	2596
Tai_50x5_4		2742	2740	2751	2740
Tai_50x5_5		2849	2839	2863	2837
Tai_50x5_6		2820	2806	2829	2793
Tai_50x5_7		2711	2692	2725	2689
Tai_50x5_8		2675	2667	2683	2667
Tai_50x5_9		2551	2543	2554	2527
Tai_50x5_10		2779	2776	2782	2776
Tai_50x10_1	50x10	3232	3022	3037	2907
Tai_50x10_2		3289	3010	2911	2821
Tai_50x10_3		2978	2867	2873	2801
Tai_50x10_4		3128	3012	3067	2968
Tai_50x10_5		3140	3014	3025	2908
Tai_50x10_6		3265	3022	3021	2941
Tai_50x10_7		3240	3078	3124	3062
Tai_50x10_8		3245	2966	3048	2959
Tai_50x10_9		3300	2920	2913	2795
Tai_50x10_10		3451	3092	3114	3046
Tai_50x20_1	50x20	4045	3785	3886	3480
Tai_50x20_2		4010	3712	3733	3424
Tai_50x20_3		3987	3678	3689	3351
Tai_50x20_4		3934	3745	3755	3336
Tai_50x20_5		4023	3648	3655	3313
Tai_50x20_6		4102	3707	3719	3460
Tai_50x20_7		4231	3698	3730	3427
Tai_50x20_8		4167	3640	3744	3383
Tai_50x20_9		4199	3743	3790	3457
Tai_50x20_10		4255	3614	3791	3438

Table-4. Makespan comparison of PFSSP benchmark instances (Size 100).

Instance Name	Instance size	Makespan		Upper bound	Lower bound
		BSA	GBSA		
Tai_100x5_1	100x5	5787	5498	5493	5437
Tai_100x5_2		5435	5272	5274	5208
Tai_100x5_3		5459	5199	5175	5130
Tai_100x5_4		5376	5087	5018	4963
Tai_100x5_5		5583	5248	5250	5195
Tai_100x5_6		5540	5122	5135	5063
Tai_100x5_7		5598	5286	5247	5198
Tai_100x5_8		5704	5121	5106	5038
Tai_100x5_9		5982	5502	5454	5385
Tai_100x5_10		5828	5301	5328	5272
Tai_100x10_1	100x10	5924	5776	5776	5759
Tai_100x10_2		5635	5362	5362	5345
Tai_100x10_3		5989	5688	5679	5623
Tai_100x10_4		6020	5832	5820	5732
Tai_100x10_5		5856	5532	5491	5431
Tai_100x10_6		5567	5310	5308	5246
Tai_100x10_7		5980	5602	5602	5523
Tai_100x10_8		5934	5670	5640	5556
Tai_100x10_9		6249	5943	5891	5779
Tai_100x10_10		6288	5899	5860	5830
Tai_100x20_1	100x20	6867	6250	6345	5851
Tai_100x20_2		6923	6245	6323	6099
Tai_100x20_3		6940	6334	6385	6099
Tai_100x20_4		6899	6215	6331	6072
Tai_100x20_5		7021	6399	6405	6009
Tai_100x20_6		7087	6424	6487	6144
Tai_100x20_7		6945	6212	6393	5991
Tai_100x20_8		7124	6590	6514	6084
Tai_100x20_9		6806	6128	6386	5979
Tai_100x20_10		7127	6457	6544	6298

5. SUMMARY AND FUTURE WORK

In this paper, two bio-inspired techniques, bacterial foraging and our hybrid GBSA proposed as an improvement to the bacterial foraging and Swarming technique, were implemented and evaluated. GBSA was designed by combining the best features of Genetic Algorithm and Bacterial Swarming techniques. In the proposed technique, exploration of search space was performed by using the swarming methodology of bacteria and the exploitation of possibly better optima within the search space was made to be done by GA strategy. Both the techniques were applied to the Permutation Flow Shop Scheduling Problem (PFSSP). Benchmark instances proposed by E Taillard were used for performance evaluation. Experimental comparisons of these proposed techniques have clearly shown the competence of the nature-inspired methods in solving permutation flow shop scheduling problems. In solving PFSSP, GBSA has

achieved nearly 78% of optimal solutions much closer to the lower bound, whereas BSA got only 26% of its solutions closer to the lower bound. There is a noticeable improvement in optimal solutions in GBSA and it has outperformed the existing bacterial swarming method in all the cases. This work can be extended by implementing the proposed technique to dynamic flow shop in real factory environment settings. Also, further research works can be carried out by varying the parameter values of both bacterial swarming technique and genetic algorithm to identify the best set of parameters that could further optimize the schedule.

ACKNOWLEDGEMENT

This research work is a part of the minor research project funded by University Grants Commission, India. The authors would like to thank the UGC for their financial support and encouragement in carrying out this research.

REFERENCES

[1] Radha Thangaraj, Millie Pant, Ajith Abraham and Pascal Bouvry. 2011. Particle swarm optimization: Hybridization perspectives and experimental illustrations. Applied Mathematics and Computation. Elsevier. doi:10.1016/j.amc.2010.12.053.

[2] Jason Brownlee. 2011. Clever Algorithms: Nature-Inspired Programming Recipes. First Edition.

[3] Wu1 C. Zhang N. Jiang J. Jinhui Yang J. and Liang Y. 2007. Improved Bacterial Foraging Algorithms and Their Applications to Job Shop Scheduling Problems. Springer-Verlag Berlin Heidelberg.

[4] Abolhasani Ashkezari M.H., Shahsavari Pour N and Mohammadi Andargoli H. 2012. An Ant Colony System for solving fuzzy flow shop scheduling problem. International Journal of Engineering and Technology. 1(2): 44-57 ©Science Publishing Corporation.

[5] Hela Boukef, Mohamed Benrejeb and Pierre Borne. 2007. A Proposed Genetic Algorithm Coding for Flow-Shop Scheduling Problems.

[6] Ashwani Kumar Dhingra. 2010. Multi-Objective Flow Shop Scheduling using Metaheuristics.

[7] Orhan Engi N and Alper Döyen. 2007. A New Approach to Solve Flow shop Scheduling Problems by Artificial Immune Systems.

[8] Weise T. 2003. Global Optimization Algorithms-Theory and Application. 2nd Edition.

[9] Mircea Ancău. 2012. On Solving Flow shop Scheduling Problems. The Publishing House of the Romanian Academy. Proceedings of the Romanian Academy. Series A. 13(1): 71-79.

[10] Seda Hezer and Yakup Kara. 2011. Solving Vehicle Routing Problem with Simultaneous Delivery and Pick-up using Bacterial Foraging Optimization Algorithm.

[11] Tang W.J. 2008. Bacterial Foraging Algorithm for Optimal Power Flow in Dynamic Environments. IEEE Transactions on Circuits and Systems.

[12] Zulkifli Zainal Abidin, Mohd Rizal Arshad and Umi Kalthum Ngah. 2009. A Survey: Animal-Inspired Metaheuristic Algorithms. Proceedings of the Electrical and Electronic Postgraduate Colloquium EEPC.

[13] Liu W. Chen H. and Chen M. 2011. RFID Network Scheduling Using an Adaptive Bacterial Foraging Algorithm. Journal of Computational Information Systems. pp. 1238-1245.

[14] Dasgupta S., Biswas A., Das S., Panigrahi B.K. and Abraham A. 2008. A Micro-Bacterial Foraging Algorithm for High-Dimensional Optimization.

[15] Kim D.H. Abraham A. and Cho J.H. 2007. A hybrid genetic algorithm and bacterial foraging approach for global optimization. Information Sciences. p. 177.

[16] Taillard E. 1989. Benchmarks for Basic Scheduling Problems.

[17] Samiakouki. Mohamed Jemni and Talel Ladhari. 2011. Solving the Permutation Flow Shop Problem with Makespan Criterion using Grids.