Indian Journal of Science and Technology, *Vol 9(4), DOI: 10.17485/ijst/2016/v9i4/80082, January 2016*
ISSN (Print) : 0974-6846
ISSN (Online) : 0974-5645

Application of Heuristics for Parallel Flow Line Scheduling Problem

S. Rajendran[1]* and K. Balasubramanian[2]

[1]Dr. M. G. R. Educational and Research Institute, University, Chennai - 600107, Tamil Nadu, India; rajendiran.s@gmail.com
[2]Department of Mechanical Engineering, Kalasalingam Academy of Research and Education, Krishnankoil - 626126, Tamil Nadu, India; bala_manu2002@yahoo.co.in

Abstract

The flowshop scheduling problems are solved by heuristic methods in view of the NP-hardness. This paper proposes to explore the near optimal sequences based on heuristic algorithms with the objective to minimize Makespan. Metaheuristic based methods are used in scheduling problems where exact methods are not sufficient to provide a solution. Swarm intelligence systems are generally made up of a population of naturally existing phenomena or agents and in this paper the consideration is the foraging behavior of honey bees. Such swarm based algorithms are used to duplicate the methods of nature to conduct a search towards the near optimal solution. This paper presents an application of parallel flow line scheduling using metaheuristic method of swarm intelligence based on bee colony algorithm. The algorithm used is to minimize the Makespan, which is one of the important requirements to reduce the overall lead time in manufacturing. The results of this algorithm has been compared with the results of other comparable research algorithms and verified. Computational results show that GA based algorithm outperforms ABC in all instances of the chosen problem sets. In order to measure the efficiency of the solution methods, the execution time (sec) taken by each solution method to obtain solution to an instance of a problem was computed. The mean value of execution time over the hundred problem instances solved under various metaheuristics shows that for the small size problems the computation time is same for all the three algorithms and when the problem size increases the computation time of algorithms also increases and vary exponentially and from the experiments it is inferred that the GA algorithm is faster than ABC. The application of this algorithm discussed can be applied over a number of applications related to manufacturing in parallel assembly lines like packaging industry, manufacturing industry like paper industry, plastics injection molding industry etc.

Keywords: Bee Colony, Heuristics, Parallel Flow-Line, Scheduling

1. Introduction

Nature inspired metaheuristic is one of the techniques to solve complex problems by utilizing natural behavior as observed in animals, birds, insects etc. This is mainly due to the insufficient classical optimization algorithms in solving large scale combinatorial problems. In addition, the problems are to be simplified and modeled in such a way that they can be solved using classical techniques, e.g. linear programming techniques. Due to the fact that all metaheuristics approaches use both randomization and local search, it becomes feasible to reach a global optimum, by moving away from the local search space. Such metaheuristic, though do not confirm or guarantee the perfect global optimum solution, but provides a near-optimum solution for most complex problems, which are

*Author for correspondence

prohibitively time consuming to reach such solution or near impossible to arrive at a satisfactory solution by any other exact solution methods. This paper describes the bee algorithm which is inspired by the foraging behavior of honey bees by artificially duplicating the known steps in their foraging behavior.

Section 2 reviews related work in the area of bee colony optimization by a survey of literature. Section 3 describes the honey bee foraging algorithm and Section 4 has the general procedure and Section 5 the results and inferences. There is a verification that the algorithm can be reliably used for similar problems considered and enables to reach the near global optimum solutions.

Dhingra and Paul[1] discussed bacterial foraging optimization for utility based IT services. Karaboga et al.[2] discussed an artificial bee colony algorithm for solving constrained optimization problems to test and compare the results with various test functions and explained that in most of the cases the ABC algorithm is reliable. Pham et al.[3] and Von F. K. et al.[4] discussed the behavior of bees during foraging, bees' algorithm and demonstrated the efficiency and robustness of the algorithm based on the results with a number of benchmark problems. Rodriguez et al.[5] discussed an unrelated parallel machine scheduling problem using Artificial Bee Colony algorithm. Rosenbloom et al.[6] and Buxey et al.[7] has discussed on the possible applications of parallel scheduling in industrial environment. Francisco et al.[4] has explored an Artificial Bee Colony algorithm type using a local search method and an iterated greedy structure procedure and verified the results from the existing literature of previous experiments. Sandeep Kumar et al.[8] discussed on a novel crossover based ABC algorithm with genetic algorithm which has been explained to be a strengthening factor for the exploitation phase of ABC. Yu-Yan Han et al.[9] discussed on an improved artificial bee colony algorithm for a flowshop scheduling problem. The search on online literature related to application of bee algorithm for parallel flow line problems in the previous research works was not successful. Herbert et al.[10] discussed a heuristic algorithm for sequencing without the use of digital computer

2. Bee Colony Algorithm

The bees algorithm is a population based search algorithm, first developed in 2005 by Pham DT et al. and Karaboga D. independently. The algorithm mimics the food foraging behavior of swarms of honey bees. A colony of bees can stretch itself over long distances (approximately 10 km) in multiple directions to search for a large number of food sources. In principle, the flower patches with plenty of nectar which can be collected with lesser effort shall be visited by more bees whereas those with lesser nectar receive fewer bees.

The foraging process starts in a colony by scout bees being sent to search for promising flower sources which are considered as a patch. Scout bees move randomly from one patch to another[3].

Normally, the bees are classified into three groups, those finding new food sources are called scout bees, those which recover the already identified food are called as employed bees and onlooker bees wait in the hives to select a food source based on the dances performed by employed bees in the dancing area.

In principle, each food source becomes a solution point while comparing the real life situation, which implies that there are as many solutions as the number of employed bees. The steps taken in the algorithm on the basis of the original proposal by Karaboga[1] is explained below:

The number of food sources is assumed to be equal to the employed bees or onlooker bees. Or otherwise all the food sources are treated as viable solution for a particular problem.

The objective is to schedule the jobs in a parallel flowline environment of relatively newer and older machines together in such a way that the completion times of the jobs is minimized while ensuring that the entire resources are effectively utilized.

$$\text{Minimise} \sum_{i=1}^{n} C_j$$

Where C_j represents the completion time of job j for the chosen scheduling problem.

The steps taken by the algorithm is explained below:

- To start with the population is initialized in the range of pre-defined upper and the lower limits. All the employed bees are allocated a position in the search space to bear a particular solution. The initialization is performed using the equation:

Let $X_i = (x_{i1}, x_{i2}, x_{i3} \ldots x_{in})$ be the i^{th} solution, the generation is by the expression,

$$X_{ij} = L_j + R * (U_j - L_j) \quad (1)$$

Where i = 1, 2, 3 …, Q and j = 1, 2, 3 …, n. Q is the population of the colony and n is the number of parameters

to be optimized by the algorithm and R is between 0 and 1. U is the upper limit and L is the lower limit.

- As explained earlier, each food source is associated with one employed bee, after the initialization each employed bee (Xi) is allotted a food source. After this allotment the bee makes changes the position of the food source as a new solution Xnw = (Xnw1, Xnw2… Xnwn). Once it finds a source it evaluates its quality. The neighboring food sources are defined by the following equation:

$$X_{nwj} = X_{ij} + r * (X_{ij} - X_{kj}) \quad (2)$$

Where I = 1, 2, 3 …, Q and j = 1, 2, 3, … m and k = {1, 2, 3 …, Q} and r is a randomly chosen index in the domain [-1, +1]. It can be seen from Equation (2) that as the difference between X_{ij} and X_{kj} decreases an optimal value is reached. When these two values become equal, there are no more changes in the position. After the new solution X_{nw}, it is compared with X_i. When X_{nw} is better, it replaces X_i and become the new member

- An onlooker bee choses a solution X_i depending on the probability factor Pi which is empirically calculated from:

$$p_i = \frac{Zi}{\sum_{i=1}^{Q} Zi}$$

Where Z_i is the fitness value of the i^{th} solution X_i.

From the above relation, the lower Makespan gives the lower greedy factor value. Following that a greedy selection is applied to the values obtained for x_i and Pi a better value between the two is selected according to their fitness.

- When all the employed bees have completed their searches and the first cycle is completed, the information regarding the quantity of the nectar (or quality of solution) is shared with the onlooker bees in the dancing area. An onlooker bee makes a probability on the quality of nectar (by the fitness values). In the bee algorithm a roulette wheel scheme is used for such probabilistic values. In such a probabilistic selection scheme, as the fitness of solutions increases, the number of onlookers attracted towards them also increases. This is the positive feedback feature of bee algorithm. When one search cycle is complete, the algorithm searches for the exhausted food sources by checking the counters and comparing it with a pre-set value defined by the "limit". If the values are the same the food source is abandoned and a new food source is searched by the scout bees and replaced with the abandoned source. This is the negative feedback feature of bee algorithm.
- The above steps are repeated and the possible solution sets are optimized till the criteria of the maximum number of cycles is met or the pre-defined error value is reached. By following the above steps a global minima is achieved by the algorithm.

3. General Procedure of Bee Colony Algorithm

Input: Initial set of job sequences.
Output: Makespan obtained.

Generated set of job sequences:

Set gen. = 1 and go to Start

Start: For (gen. = 1 to gen.; maxgen; gen.++)
For every job sequence
Apply shift neighborhood
Calc makespan
End for
Calculate probability of job sequence
Onlooker bee finds the best seq. for Makespan
Memorize optimal job sequence for Makespan
Scout bees randomly find new job sequence
Create new job seq. within the neighborhood
Update and generate new job seq. for gen.++
End for

4. Experiments and Results

One set of manual calculations was also done to verify a simple random problem and then a computer code using .net environment under windows platform has been generated to enable us to run multiple problem sequences by varying the parameters. An initial list of solutions is used to denote forager bees that are performing a dance on the floor. Each solution contains the foraging list, Makespan, maximum number of iterations and the acceptance point when a solution is added to the list. After every cycle, the list is updated with new solutions based on the performance criteria set.

A total of 10 replication runs are performed for each problem sequence to obtain the average of results. The total population of sequences would be n! jobs considered in terms of the four machines, four parallel lines and five jobs in sequence by applying the operators of swap, insertion and reversion. A sample set of results of the random ten sequences are tabulated below on the application of the operators during the iterations.

The input generated based on a random generation for the 4 machines, 5 jobs and 4 lines. The input used was as follows:

Table 1. Makespan over generations

J	L	M1	M2	M3	M4
J1	L1	3	4	5	7
	L2	2	4	4	5
	L3	1	3	3	4
	L4	1	2	3	4
J2	L1	7	2	1	2
	L2	6	6	1	1
	L3	5	5	6	2
	L4	4	4	5	7
J3	L1	3	4	5	6
	L2	1	2	3	4
	L3	1	1	3	4
	L4	7	1	2	2
J4	L1	5	6	1	2
	L2	5	6	6	1
	L3	4	4	5	6
	L4	2	4	5	6
J5	L1	2	2	4	4
	L2	2	1	3	3
	L3	7	1	2	3
	L4	5	6	2	1

The quantity of jobs considered for each job item is different which is tabulated as below:

Table 2. Input of jobs

J	q
J1	4
J2	5
J3	3
J4	5
J5	4

Cumulative fitness function $\Sigma F(X) = 0.00523$

Utilizing the input the above seed sequence was derived in terms of the jobs and lines and the calculations of Makespan with a check of greedy factor leading to probability of fitness was calculated. The seed sequence was subjected to the bee swarm operators at single point swap and insertions. For example in the table the first sample has the sequence 4-4 and 5-4 which has been reversed in the sample 2 as 5-4 and then followed by 4-4. Based on the probability of fitness level, the minimum make span occurred has been considered as the global optimum for the chosen problem.

Table 3. Seed sequence with Makespan computation

Seed sequence	Makespan (z)	Greedy factor (Pi)
1-1-2-2-3-3-**4**-4-**5**-4	185	0.00535
1-1-2-2-3-3-**5**-4-**4**-4	195	0.00510
1-1-2-2-**4**-3-**3**-4-5-4	200	0.00498
1-1-3-2-4-3-**5**-4-**2**-4	206	0.00483
1-1-4-2-2-3-3-4-5-4	203	0.00490
1-1-4-2-5-3-3-4-2-4	211	0.00472
1-1-5-2-4-3-2-4-3-4	203	0.00490
2-1-1-2-4-3-5-4-3-4	191	0.00521
2-1-5-2-3-3-1-4-4-4	150*	0.00667
2-1-4-2-5-3-1-4-3-4	168	0.00592
3-1-2-2-5-3-4-4-1-4	200	0.00498

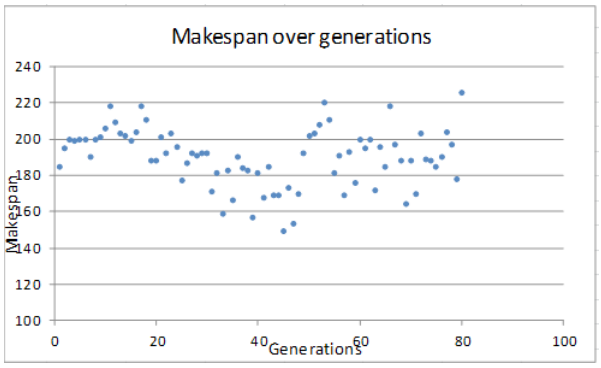

Figure 1. Makespan over generations using ABC.

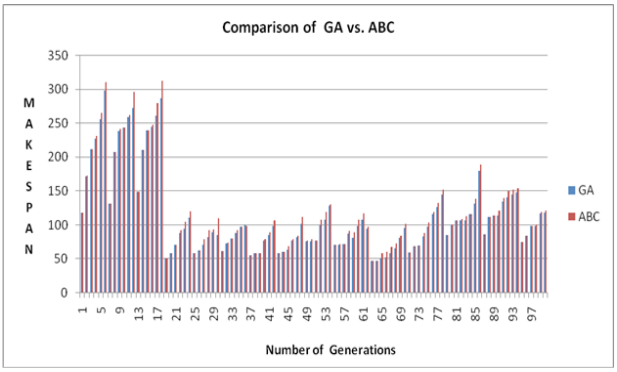

Figure-2. Comparison of GA vs. ABC.

Based on the completed procedure, the sequence 4-2-1-5-2-3-3-1-4-4-4 is considered to have a global optimal solution with a Makespan of 150 for the problem chosen with deterministic values

The graphical plot of the Makespan values over generations is as plotted below:

5. Conclusion

The study was conducted to apply the bee colony algorithm to the chosen problem with the objective of finding the minimum Makespan in a parallel flowline environment. The applications are projected in the areas of batch line production environment where similar products are to be produced in a constrained period of time and by effectively utilizing the combination of old and newer machines for the similar operation sequences. The results were also verified in terms of exhaustive calculations performed manually to ensure the reliability of the calculations for further use in similar such problems.

6. Abbreviation and Acronyms

Few abbreviations were used in the paper which are listed below for reference:

gen. – Generations
seq. - Sequences
maxgen. – Maximum number of generations
calc. – calculate
GA – Genetic Algorithm
ABC – Artificial Bee Colony Algorithm

7. References

1. Dhingra A, Paul S. Green cloud: Heuristic based BFO technique to optimize resource allocation. Indian Journal of Science and Technology. 2014 May; 7(5):685–91.
2. Karaboga D, Basturk B. Artificial Bee Colony (ABC) optimization algorithm for solving constrained optimization problems. Foundations of Fuzzy Logic and Soft Computing. Berlin Heidelberg Springer-Verlag. 2007; 4529:789–98.
3. Pham DT, Ghabarzadeh A, Koc E, Otri S, Rahim S, Zadi M. The bees algorithm – A novel tool for complex optimisation problems. Proceedings of the 2nd International Virtual Conference on Intelligent Production Machines and Systems; 2006. p. 454–9.
4. Von Frisch K. Bees: Their vision, chemical senses and language. (Revised Edition). N.Y., Ithaca, Cornell University Press: 1976.
5. Rodriguez FJ, Garcia-Martinez C, Blum C, Lozano M. An Artificial Bee Colony algorithm for the unrelated parallel machines scheduling problem. Parallel Problem Solving from Nature. Berlin Heidelberg, Springer-Verlag. 2012; 7492:143–52.
6. Rosenbloom ES, Goertzen NF. Cyclic nurse scheduling. European Journal of Operational Research. 1987 Jul; 31(1):19–23.
7. Buxey G. Production scheduling: Practice and theory. European Journal of Operational Research. 1989 Mar; 39(1):17–31.
8. Kumar S, Sharma VK, Kumari R. A novel hybrid crossover based Artificial Bee Colony algorithm for optimization problem. International Journal of Computer Applications (0975-8887). 2013 Nov; 82(8):18–25.
9. Han YY, Pan QK, Li JQ, Sang HY. An improved Artificial Bee Colony algorithm for the blocking flowshop scheduling problem. International Journal of Advanced Manufacturing Technology. 2012 Jun; 60(9):1149–59.
10. Campbell HG, Dudek RA, Smith MI. A heuristic algorithm for the N-job, M-machine sequencing problem. Management Science. 1970 Jun; 16(10):630–7.