ISSN 2347 - 3983

Volume 8. No. 1.2, 2020

International Journal of Emerging Trends in Engineering Research

Available Online at http://www.warse.org/IJETER/static/pdf/file/ijeter2581.22020.pdf

https://doi.org/10.30534/ijeter/2020/2581.22020

Manipulating the Onlooker Bee's Behaviour in Artificial Bee Colony Algorithm for Permutation Flowshop Scheduling

Salleh Ahmad Bareduan[1], Nur Fazlinda M. Pauzi[1], Noor Azizah Sidek[1], Azli Nawawi[1]

[1]Faculty of Mechanical and Manufacturing Engineering, Universiti Tun Hussein Onn Malaysia, 84600 Batu Pahat, Johor, Malaysia, saleh@uthm.edu.my, hd110165@siswa.uthm.edu.my, noorazizah@uthm.edu.my, azle@uthm.edu.my

ABSTRACT

The artificial bee colony (ABC) algorithm has attracted interest among many researchers in optimization studies. This fact is also true in the field of production planning and scheduling. This paper presents a detail step-by-step methodology to understand the effect of different onlooker bee's (OB) behaviour or strategies towards their optimization capability in solving permutation flowshop scheduling problems. This paper investigated three OB exploiting behaviours or strategies, namely total greedy, semi greedy, and non-greedy while executing the ABC algorithm in solving PFSP. Based on the simulation experiment, the authors can conclude that the exploiting behaviour of OB influences the overall ABC scheduling performance. The total greedy behaviour is the best behaviour amongst all because it generated the least error. The study also considered the total greedy behaviour as the fastest in generating 0% error data compared to other OB behaviours.

Key words: Artificial Bee Colony Algorithm, Flowshop Scheduling, Scheduling Optimization

1. INTRODUCTION

One of the most studied problems in scheduling literature is the permutation flowshop scheduling problem (PFSP) [1]. This specific scheduling problem involves searching for the best solution in scheduling n jobs that have to m machines will process the job with the restriction of the same job order on every machine. In solving the PSFP, researchers have used many different performance measures such as makespan, flow time, and tardiness [2], [3]. The total tardiness approach considers the due dates as the most critical factors in satisfying the customers and therefore is very much suitable for make-to-order manufacturing industries [4]. This approach minimized the effect of delays since delays may cause a lousy reputation, loss of customers, and cost of the penalty, as stipulated in some sales agreement. On the other hand, makespan, which measures the completion time of a group of products, is very much related to the fast processing of the products and balanced use of resources. These processing performances and resources utilization is a critical strategic criterion in managing make-to-stock manufacturing industries [5].

The PFSP is classified to be NP-hard. Therefore, instead of searching for an exact solution, many researchers focused on developing heuristics that are capable of providing a good solution. However, these heuristics do not guarantee the achievement of the optimum solution. Nevertheless, the excellent solution, which is usually close to the optimum, can be obtained within a reasonable time interval. One of the most popular and successful heuristic to solve the permutation flowshop problem is the NEH heuristic [6]. Many researchers have introduced some modifications to the NEH [7] and even used it as the initial solution for newly developed heuristics.

In the last few years, the trend has changed into using swarm intelligence concept or a specific type of genetic algorithm [8], [9] to solve the flowshop scheduling problem. One of the methods that used the swarm intelligence concept is the Artificial Bee Colony (ABC). Karaboga et al. introduced the ABC algorithm in 2005, and it was used as a medium to optimize multivariable and multimodal continuous functions [10]. Three types of artificial bees known as employed bees (EB), onlooker bees (OB), the method sent scout bees (SB) to find food sources under the definition of the task of exploitation and exploration [11]. The ABC algorithm has now received very significant attention among researchers, and the community utilizes it in solving many optimization problems. Previous researchers reported the application of the ABC algorithm in searching the optimum solution for numerical function problems [12] and also for solving the lot-streaming flowshop scheduling problem [13]. Other reports introduced a Pareto-based ABC algorithm to investigate multi-objective flexible job-shop scheduling problems, ABC clustering model in protein-protein interaction networks based on a propagating mechanism, a feature selection technique based on ABC for image steganalysis problems and the utilization of ABC to design the intelligent PID/PIλDμ speed controller for chopper fed DC motor drive [14]–[18]. A paper also reported a straightforward application of the ABC algorithm to solve PFSP, and it also compares the ABC performance against the conventional dispatch rules [19]. The ABC was also used for optimizing total tardiness in the no-idle permutation flowshop scheduling problem [20] and applied in constrained optimization problems [21]. Currently, researchers studied ABC by introducing a hybrid approach in the intention to obtain better performance solutions. This study includes the proposed hybrid algorithm artificial bee colony algorithm with some steps of genetic algorithm to achieve Pareto solutions for multi-objective single machine group scheduling problem with sequence-dependent setup times and learning effects [14]. A paper also reported the hybrid ABC combining bees approach and a deep level local search mechanism for solving the multi-objective flexible task scheduling problem in the Cloud computing system [22]. In another study, a research team experimented on a combination of genetic programming and artificial bee colony algorithm to get a better balance between exploration and exploitation leading to a mechanism proposed to attract individuals towards a promising solution region [23]. The hybrid effort was extended further in the area of unrelated parallel machine scheduling problem with deteriorating maintenance activities, parallel-batching processing, and deteriorating jobs (a combination of the artificial bee colony and Tabu Search) to solve the problem in a reasonable time [24].

However, from the literature, previous works gave not much attention in conducting detail investigation to understand the effect of different bee's behaviours or strategies towards their optimization capability. Therefore, in this work, several manipulations on the onlooker bee's behaviour were proposed, and our team also investigated its effect in solving permutation flow-shop scheduling problems with unlimited buffers.

The permutation flowshop scheduling finds the best permutation to minimize the maximum completion time or makespan. The permutation of n jobs represents the solution to the permutation flowshop scheduling problem, and the system has a set of n jobs, $\pi = \pi1, \pi2, \ldots, \pi n$. The system processes each job on m operations. Different machines will perform every operation and the processing time pji for job j using machine i is given. This method will find the best permutation for jobs $\pi^* = \{\pi1^*, \pi2^*, \ldots, \pi n^*\}$ to be processed on each machine using the permutation flowshop scheduling. Let, C (πj,m) denotes the completion time for the job πj using machine m. Given the job permutation π, the completion time for the n job, m machine problem is calculated as follows:

$$C(\pi1, 1) = P\pi1, 1 \quad (1)$$
$$C(\pi j, 1) = C(\pi j - 1, 1) + P\pi j,1 \; j = 2, \ldots, n \quad (2)$$
$$C(\pi1, i) = C(\pi1, i - 1) + P\pi1,i \quad i = 2, \ldots, m \quad (3)$$
$$C(\pi j, i) = \max[C(\pi j - 1, i), C(\pi j, i - 1)] + P\pi j,i; \; j=2, \ldots, n; i = 2, \ldots, m \quad (4)$$

The makespan for a permutation π is equal to the completion time for the last job πn using the last machine m. The completion time for the permutation π is Cmax(π)=C(πn, m). Other characteristics of the permutation flowshop are as the following:

i. Each job visits each stage according to the same production flow.
ii. In every stage i, there is only one machine with specified processing abilities.
iii. Between the stages i and i + 1, there are unlimited buffer capacities.
iv. Every machine will process only one job at a time, and each job can be executed by only one machine at a time.
v. All jobs and machines are available at the initial of the process (t=0).
vi. Preemption is not allowed; that is, a job cannot be interrupted before the completion of its current operation.
vii. The method will include the setup times in the processing time, and the problem data are deterministic and known in advance.

The paper aims to generate the schedule sequence to minimize the makespan.

2. MATERIALS AND METHODS

The artificial bee colony algorithm system starts with the movement of an SB to find the food source randomly. Food source found by SB is known as the initial solution. In the flowshop scheduling situation, the SB will select any scheduling sequence randomly. The quality of this initial solution is measured using the makespan value. Using the initial solution as a guide, a few EB start searching for new alternative solutions around the initial solution. The method evaluated the quality of these alternative solutions by calculating their respective makespan value.

Upon comparing the makespan from the EB solutions, the OBs have to decide on which solution to choose as a guide for further solution exploitation. In this paper, our team proposed that the OB have three different selecting behaviours as the following:

i. Total greedy behaviour: All OB select only the best EB solution as a searching guide.
ii. Semi greedy behaviour: 67%OB select the best EB solution as a searching guide. The remaining 33% OB select the second best EB solution as the alternative searching guide.
iii. Non-greedy behaviour: Each OB selects one EB solution as a searching guide.

The makespan from all EB and OB are compared to identify the best solution. If the best solution is better than the initial solution, then the solution is kept as the optimized result and is used as a guide for the next EB activity. If not, an SB is released to find a new initial solution randomly. The best solution among all bees is considered as the optimized solution. The simulation will repeat the process until it achieves the termination limit. The developed artificial bee colony algorithm had been set to solve the permutation flowshop scheduling problem for six jobs and three machines. Table 1 shows a randomly generated process time data for a three machine flowshop problem.

Table 1: The example of process time data for flowshop (hours)

Job	M1	M2	M3

A	22	11	16
B	2	6	7
C	36	10	29
D	8	28	17
E	27	1	14
F	10	7	40

The ABC starts with the movement of an SB to find the initial solution randomly. The selected initial solution is DCFAEB with makespan equals 160, and the next steps are the movement of EB, followed by OB. All movements of both types of bees are counted as iterations. In this example problem, three EB and three OB are utilized in each cycle to make a total count of six repetitions per period. After receiving the information about the initial solution, three EB search the new alternative solutions nearby the area of the initial solution using the select and insert method. The EB sequence solutions are shown in Table 2.

The OB uses the information from EB in Table 2 to decide which EB solution to be selected as the guide for the next movement. The guide selection is based on the three OB behaviours as the following:

i. Total greedy behaviour: All three OB select the best EB solution, which is DFCAEB from EB3 as the searching guide.
ii. Semi greedy behaviour: Two OB select the best EB solution, which is DFCAEB from EB3 as the searching guide. The remaining one OB selects the second-best EB solution, which is DACFEB from EB2 as the alternative searching guide.
iii. Non-greedy behaviour: Each OB selects one EB solution as the searching guide.

Table 2: The sequence of solutions of EB from the first cycle

EB	Sequence Solutions	Makespan
EB1	DECFAB	173
EB2	DACFEB	166
EB3	DFCAEB	159

Semi greedy behaviour is chosen in solving the example problem, and this resulted in the OB solutions shown in Table 3.

Table 3: The sequence solutions of OB from the first cycle

Selected EB Guide	OB	Sequence Solutions	Makespan
EB3 (DFCAEB)	OB1	DFCBAE	159
EB3 (DFCAEB)	OB2	DFCBEA	159
EB2 (DACFEB)	OB3	DACBFE	166

The best makespan from all EB and OB solutions in the first cycle is 159, and this is better than the initial solution makespan of 160. Therefore, the best solution among EB and OB is kept as the current optimized solution and selected as the guide or sub-initial for the next EB searching activity. In this example, DFCBEA is chosen as the current optimized solution and the sub-initial for the following cycle.

Using the new sub-initial solution as the guide, three EB search the new alternative solutions nearby the sub-initial solution area using select and insert method. The EB sequence solutions are shown in Table 4.

Table 4: The sequence solutions of EB from the second cycle

EB	Sequence Solutions	Makespan
EB1	DEFCBA	159
EB2	DBFCEA	159
EB3	DCFBEA	160

Applying the semi greedy behaviour, two OB select the best EB solution, which is DEFCBA from EB1 as the searching guide. The remaining one OB selects the second-best EB solution, which is DBFCEA from EB2 as the alternative searching guide. This situation resulted in the OB solutions shown in Table 5.

Table 5: The sequence solutions of OB from the second cycle

Selected EB Guide	OB	Sequence Solutions	Makespan
EB1 (DEFCBA)	OB1	DEFACB	159
EB1 (DEFCBA)	OB2	DEFABC	159
EB2 (DBFCEA)	OB3	DBFACE	159

The best makespan from all EB and OB solutions in the second cycle is 159, and this is not better than the currently optimized solution makespan of 159 obtained from the first cycle. Therefore, an SB is released to find a new initial solution randomly. The latest initial solution selected is BDCEAF with makespan of 162. Using this new initial, the EB and OB repeat their searching activity following the semi greedy behaviour resulting in the sequence solution shown in Table 6.

Table 6: EB and OB sequence solutions from the third cycle

EB Sequence Solutions	Makespan	OB Sequence Solutions	Makespan
EB1 (BADCEF)	163	OB1 (BADFCE)	163
EB2 (BEDCAF)	168	OB2(BADFEC)	163
EB3 (BCDEAF)	164	OB3 (BCDFEA)	164

The best makespan from all EB and OB solutions in the third cycle is 163, and this is not better than the currently optimized solution makespan of 159 obtained from the first cycle. Therefore, an SB is rereleased to find a new initial solution randomly. Since these three machines flowshop examples problem uses three EB and three OB in its cycle, therefore every period adds six alternative solutions or number of iterations to its EB and OB exploitation activities counter. The overall searching process stops only when a predetermined stopping criterion is met. In this example problem, the stopping criterion is set at 102 iterations. The result upon executing the 102 iterations is shown in Table 7.

Table 7 depicted that there are several occasions in which the cycle solutions are better than the current optimized solution. This scenario happened at EB and OB makespan values of 159, 144, 140 and 135. Finally, after 102 iterations, the best solutions among all EB, OB and SB are BFDACE with makespan value of 135.

This study evaluated the quality of the ABC solution by comparing the makespan resulted from the ABC solution with the actual optimum solution of the example flowshop problem. The exact optimum solution is obtained by computing the makespan values for all possible sequence arrangements. Since the example problem involves six jobs (n=6), therefore a total of n! or 720 different sequence arrangements have to be investigated to search for the minimum makespan. This situation has resulted in an optimum solution makespan of 135. Therefore, for the example problem discussed in this section, the ABC managed to obtain a solution makespan equals to the optimum solution. This situation resulted in a zero per cent makespan error by using Equation (5).

$$Makespan\ Error\ (\%) = \left(\frac{Makespan\ for\ ABC - Optimum\ Makespan}{Optimum\ Makespan}\right) \times 100 \quad (5)$$

3. RESULTS AND DISCUSSIONS

As a way to have a better understanding of the performances of the ABC algorithm with different OB behaviours, in solving flowshop scheduling problems, computational simulations resembling a three ma-chine flowshop scheduling were conducted. One hundred sets of randomly generated process time data for the flowshop were used in the analysis. For each data set, the quality of the ABC result was measured using the makespan error percentage as in Equation (5). Upon completion of all one hundred sets of simulation, the overall average makespan error was computed to determine the overall performance of the OB behaviours. Since the ABC process involves selecting a random initial solution, the whole simulation was repeated with ten replications. The average makespan error from all replications was used as the performance of the ABC algorithm in solving the flowshop scheduling problems. The simulation process was executed in Microsoft Excel with Visual Basic Application using the termination criterion of 54, 102, 204, 300 and 402 iterations. The result is shown in Table 8.

Table 8: Average makespan error of the ABC algorithm (%)

OB Behavior	Number of Iterations				
	54	102	204	300	402
Total Greedy	1.511	0.569	0.269	0.199	0.119
Semi Greedy	2.096	0.708	0.362	0.249	0.136
Non-Greedy	2.190	0.743	0.409	0.262	0.234

The result in Table 8 shows that as the number of iterations increased, the performances of the ABC algorithm also increased. This scenario is indicated by the lower makespan error value as the number of iterations increases. A higher number of iterations means more alternative solutions are explored and exploited by SB, EB and OB. Therefore, there is a higher chance that a better solution is found. In order to have a more precise observation, the data in Table 8 is tabulated into the interaction plot, as illustrated in Figure 1.

Table 7: ABC results at 102 iterations

Initial Solution						Initial Makespan	MinEBOB						Makespan	Iteration Counter
D	C	F	A	E	B	160	D	F	C	B	E	A	159	6
							D	B	F	A	C	E	159	12
B	D	C	E	A	F	162	B	A	D	F	E	C	163	18
F	A	D	E	C	B	149	F	D	A	B	E	C	144	24
							F	B	D	C	A	E	140	30
							F	D	B	C	A	E	140	36
F	D	C	B	E	A	140	F	C	D	A	B	E	140	42
C	F	D	B	A	E	169	C	B	F	E	D	A	169	48
E	D	A	F	C	B	172	E	F	D	B	C	A	153	54
E	F	A	D	B	C	153	E	B	F	C	D	A	151	60
F	E	D	A	B	C	144	F	D	E	C	A	B	143	66
D	F	B	A	E	C	159	D	A	F	C	B	E	159	72
A	F	D	B	E	C	156	A	B	F	C	E	D	156	78
D	F	C	E	A	B	159	D	E	F	B	C	A	159	84
C	E	B	D	F	A	174	C	D	E	A	B	F	169	90

F	D	B	E	C	A	140	F	C	D	A	E	B	140	96
B	D	F	E	C	A	154	B	F	D	A	C	E	135	102

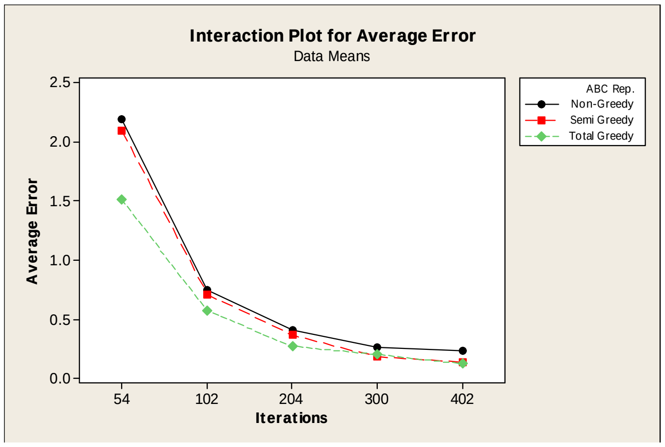

Figure 1: Interaction plot for average makespan error (%)

Table 8 and Figure 1 also show that at all iterations, the total greedy behaviour produces the lowest error percentage followed by semi greedy behaviour. Additionally, it can also be noticed that at 300 iterations, all OB behaviours seems to generate almost equal error percentage values. This pattern is also supported by Figure 2.

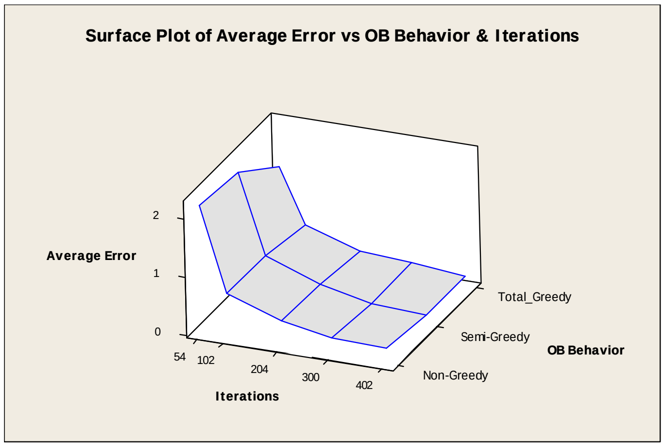

Figure 2: Surface plot shows the pattern generated by all OB behaviours

In the total greedy behaviour, the OB focused all their efforts toward looking for alternative solutions near one dedicated area, which is the best EB solution. In the semi greedy behaviour, the OB searching efforts are focused at the best and second-best EB solutions, whereas in non-greedy behaviour, no extra focus is given to reasonable EB solutions. As a result, the non-greedy behaviour recorded the worst performance at all five iteration clusters, as shown in Table 8, Figure 1 and Figure 2. This situation means that failure to provide sufficient attention in exploiting good EB solutions will not offer good overall results.

Figure 2 shows that the total greedy behaviour has the lowest curve, and this shows that this behaviour managed to generate the lowest percentage of average error, which is the desired result. In other words, the lowest curve means less error generated. Moreover, it can also be concluded that all OB behaviours demonstrated a similar pattern. As the number of iterations increased, the values of the average error will decrease.

Figure 3 provides more detailed results based on the simulation experiment. The figure shows the number of data generated based on the error ranges, iterations and OB behaviour. The error ranges are divided into five categories:

i. 0% error.
ii. 0.1% to 5.0% error range.
iii. 5.1% to 10.0% error range.
iv. 10.1% to 15.0% error range.
v. 15.1% to 20.0% error range.

The OB behaviours are represented by different colours. The non-greedy, semi greedy and total greedy behaviours are represented by the red, blue and green colours respectively. From Figure 3, the first category of the error range (0% error) is the best because the generated results have the same value as the optimum makespan.

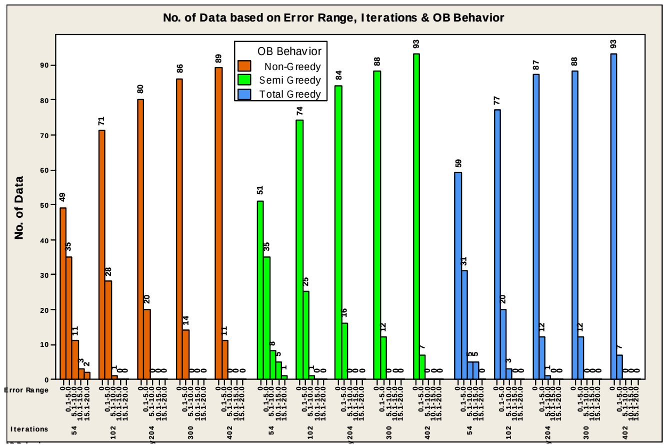

Figure 3: No. of data based on error ranges, iterations and OB behaviour

Besides, it can also be seen that all OB behaviours produced the same trend. All behaviours tend to generate more data that fall in the 0% error category as the number of iterations increased. However, it was also observed that the total greedy behaviour has the highest rate in generating 0% error data because the amount of data that fall into this category is the highest for all iterations (54, 102, 204, 300 and 402). From here, it can be concluded that the best OB behaviour is the total greedy because it managed to generate the least error.

4. CONCLUSION

This study successfully presented a detail step-by-step methodology to investigate three OB exploiting behaviour or strategies while executing the ABC algorithm in solving the flowshop scheduling problem. Based on the simulation experiment, it is concluded that OB-exploiting behaviour influences the overall ABC scheduling performance. Based on this study, the total greedy behaviour is the best behaviour amongst all because it generated the least error. The full greedy behaviour is also considered as the fastest in generating 0% error data compared to other OB behaviours. The findings from this study can trigger many different alternative methods to manipulate the bees' behaviour as a means of intelligent element for optimization mechanisms.

ACKNOWLEDGEMENT

The authors would like to thank the Ministry of Education Malaysia for supporting this research under the Fundamental Research Grant Scheme VotNo.FRGS/1/2018/TK03/ UTHM/02/9 and partially sponsored by Universiti Tun Hussein Onn Malaysia.

REFERENCES

1. R. Ruiz and C. Maroto, **A comprehensive review and evaluation of permutation flowshop heuristics**, in *European Journal of Operational Research*, vol. 165, no. 2, pp. 479–494, Sep. 2005.
2. J. M. Framinan, J. N. D. Gupta, and R. Leisten, **A Review and Classification of Heuristics for Permutation Flow-Shop Scheduling with Makespan Objective**, *The Journal of the Operational Research Society*, vol. 55, no. 12, pp. 1243–1255, 2004.
3. J. Schaller, **Scheduling a permutation flow shop with family setups to minimise total tardiness**, *International Journal of Production Research*, vol. 50, no. 8, pp. 2204–2217, Apr. 2012.
4. Y.-D. Kim, J.-G. Kim, B. Choi, and H.-U. Kim, **Production scheduling in a semiconductor wafer fabrication facility producing multiple product types with distinct due dates**, *IEEE Transactions on Robotics and Automation*, vol. 17, no. 5. pp. 589–598, 2001.
5. V. Fernandez-Viagas and J. M. Framinan, **NEH-based heuristics for the permutation flowshop scheduling problem to minimise total tardiness**, *Computers and Operations Research*, vol. 60, pp. 27–36, 2015.
6. M. Nawaz, E. E. Enscore, and I. Ham, **A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem**, *Omega*, vol. 11, no. 1, pp. 91–95, 1983.
7. V. Fernandez-Viagas and J. M. Framinan, **On insertion tie-breaking rules in heuristics for the permutation flowshop scheduling problem**, *Computers and Operations Research*, vol. 45, pp. 60–67, 2014.
8. X. Liu and T. P. Chung, **A modified immunoglobulin-based artificial immune system algorithm for solving the permutation flow shop scheduling problem**, *Journal of Industrial and Production Engineering*, vol. 34, no. 7, pp. 542–550, Oct. 2017.
9. C. Ou-Yang and R. Ansari, **Applying a hybrid particle swarm optimization_Tabu search algorithm to a facility location case in Jakarta**, *Journal of Industrial and Production Engineering*, vol. 34, no. 3, pp. 199–212, Apr. 2017.
10. D. Karaboga, **An idea based on Honey Bee Swarm for Numerical Optimization**, *Technical Report TR06, Erciyes University*, no. TR06, p. 10, 2005.
11. D. Karaboga and B. Basturk, **On the performance of artificial bee colony (ABC) algorithm**, *Applied Soft Computing Journal*, vol. 8, no. 1, pp. 687–697, 2008.
12. D. Karaboga and B. Basturk, **A powerful and efficient algorithm for numerical function optimization: Artificial bee colony (ABC) algorithm**, *Journal of Global Optimization*, vol. 39, no. 3, pp. 459–471, Nov. 2007.
13. M. F. Tasgetiren, Q. K. Pan, P. N. Suganthan, and A. H. L. Chen, **A discrete artificial bee colony algorithm for the permutation flow shop scheduling problem with total flowtime criterion**, 2010.
14. L. Yue, Z. Guan, U. Saif, F. Zhang, and H. Wang, **Hybrid Pareto artificial bee colony algorithm for multi-objective single machine group scheduling problem with sequence-dependent setup times and learning effects**, *SpringerPlus*, vol. 5, no. 1, Dec. 2016.
15. J. Q. Li, Q. K. Pan, and K. Z. Gao, **Pareto-based discrete artificial bee colony algorithm for multi-objective flexible job shop scheduling problems**, *International Journal of Advanced Manufacturing Technology*, vol. 55, no. 9–12, pp. 1159–1169, Aug. 2011.
16. X. Lei, J. Tian, L. Ge, and A. Zhang, **The clustering model and algorithm of PPI network based on propagating mechanism of artificial bee colony**, *Information Sciences*, vol. 247, pp. 21–39, 2013.
17. F. G. Mohammadi and M. S. Abadeh, **Image steganalysis using a bee colony based feature selection algorithm**, *Engineering Applications of Artificial Intelligence*, vol. 31, pp. 35–43, 2014.
18. A. Rajasekhar, R. Kumar Jatoth, and A. Abraham, **Design of intelligent PID/PIλDμ speed controller for chopper fed DC motor drive using opposition based artificial bee colony algorithm**, *Engineering Applications of Artificial Intelligence*, vol. 29, pp. 13–32, 2014.
19. H. Y. Chow, S. Hasan, and S. A. Bareduan, **Basic concept of implementing artificial bee colony (ABC) system in flow shop scheduling**, in *Applied Mechanics and Materials*, vol. 315, pp. 385–388, 2013.
20. M. Fatih Tasgetiren, Q. K. Pan, P. N. Suganthan, and A. Oner, **A discrete artificial bee colony algorithm for the**

no-idle permutation flowshop scheduling problem with the total tardiness criterion, *Applied Mathematical Modelling*, vol. 37, no. 10–11, pp. 6758–6779, 2013.

21. H. C. Tsai, **Integrating the artificial bee colony and bees algorithm to face constrained optimization problems**, *Information Sciences*, vol. 258, pp. 80–93, 2014.
22. J. Q. Li, Y. Y. Han, and C. G. Wang, **A Hybrid Artificial Bee Colony Algorithm to Solve Multi-objective Hybrid Flowshop in Cloud Computing Systems**, in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 10602, pp. 201–213, 2017.
23. D. Li, R. Guo, R. Zhan, and Y. Yin, **An innovative artificial bee colony algorithm and its application to a practical intercell scheduling problem**, *Engineering Optimization*, vol. 50, no. 6, pp. 933–948, Jun. 2018.
24. S. Lu, X. Liu, J. Pei, M. T. Thai, and P. M. Pardalos, **A hybrid ABC-TS algorithm for the unrelated parallel-batching machines scheduling problem with deteriorating jobs and maintenance activity**, *Applied Soft Computing Journal*, vol. 66, pp. 168–182, 2018.