International Journal on Advanced Science Engineering Information Technology

Vol.12 (2022) No. 6
ISSN: 2088-5334

A Solving Route Optimization of Airplane Travel Problem Use Artificial Bee Colony Algorithm

I Gusti Agung Premananda [a], Ahmad Muklason [a,*], Rizal Risnanda Hutama [a]

[a] *Department of Information System, Institut Teknologi Sepuluh Nopember, Sukolilo, Surabaya, 60111, Indonesia*
Corresponding author: [*]mukhlason@is.its.ac.id, ahmad.muklason@gmail.com

***Abstract*— The Traveling Salesman Problem (TSP) is very popular in combinatoric optimization. The TSP problem is finding the optimal route from several cities where the distance between cities is known, and a salesman must visit each city exactly once and return to the origin city. The goal is to find a route with a minimum total distance. This problem is known as a non-deterministic polynomial hard (NP-hard) problem, which means the computation time to find a solution increases exponentially with the size of the problem. NP-Hard problems can be solved by using heuristic methods where the solution obtained is good enough (does not guarantee the most optimal solution) in a reasonable time. One of the most recent variants of TSP problem is finding the cheapest flight routes to several cities, which is part of the Traveling Salesman Challenge 2.0 (TSC 2.0) 2018 competition . This paper reports our study of implementing an artificial bee colony (ABC) algorithm for the TSC 2.0 problem. ABC algorithm is chosen based on its superiority over other algorithms in several optimization problems. The algorithm is implemented in a hyper-heuristic form. Several combinations of swap operators are used to find the best combination result. The experimental result shows that the ABC algorithm can solve the TSC 2.0 problem with a fairly good performance by producing a savings cost of 54.6% from the initial solution and 26% compared to the Genetic Algorithm.**

***Keywords*— Traveling salesman problem; artificial bee colony; traveling salesman challenge 2.0.**

Manuscript received 27 Nov. 2021; revised 24 Apr. 2022; accepted 6 Jun. 2022. Date of publication 31 Dec. 2022.

I. INTRODUCTION

TSP is one of the most popular problems in the world of combinatoric optimization [1]–[3]. Euler documented TSP in 1759, whose interest was solving the knight's tour problem [4]. In the TSP problem, it is known the number of cities and the distance between them. A salesman will start from one of the cities and depart to visit all cities exactly once and return to his hometown [5]. This problem's goal is finding the right order of cities to cover the minimum distance possible [6]. TSP can be described in a mathematical model as in the equation below:

$$X_{ij} \begin{cases} 1 \text{ the path goes from city i to city j} \\ 0 \text{ otherwise} \end{cases} \quad (1)$$

$$\min \sum_{i=0}^{n} \sum_{j \neq i, j=0}^{n} C_{ij} X_{ij} \quad (2)$$

$$0 \leq X_{ij} \leq 1 \qquad i, j = 0, \dots, n \quad (3)$$

$$\sum_{i=0, i \neq j}^{n} X_{ij} = 1 \quad j = 0, \dots, n \quad (4)$$

$$\sum_{i=0, i \neq j}^{n} X_{ij} = 1 \quad i = 0, \dots, n \quad (5)$$

$$u_i \in \boldsymbol{Z} \qquad i = 0, \dots, n \quad (6)$$

$$u_i - u_j + n x_{ij} \leq n - 1 \quad 1 \leq i \neq j \leq n \quad (7)$$

Equation 1 is the decision variable of the TSP problem. X_{ij} is equal to 1 if city i is connected to city j, and X_{ij} is equal to 0 if city i and city j are not connected [7]. Equation two explains the objective function of TSP, which is to find the cheapest cost to travel in a number of cities. The objective function is obtained by multiplying C_{ij} which means the cost from city i to city j and X_{ij} which is the decision variable [8]. Equations 3 to 8 are constraints that must be met in the TSP problem[9].

TSP is also one of the most difficult combinatorial optimization problems. The TSP problem is included in the NP-Complete problem [10]–[12] where the running time increases exponentially with the increase in the size of problem [13]. Therefore, the exact algorithm's ability to solve this problem depends on the size of the problem. If the size increases, it is possible that this problem cannot be solved in an acceptable time [14]. One type of algorithm that can produce a solution quickly and the resulting solution is quite

good is a heuristic algorithm [15]. Previous research has proven the effectiveness of heuristic algorithms in solving NP-Complete problems: Ha et al [16] apply a hybrid genetic algorithm (GA) to the TSP problem with drones, de Freitas and Penna [17] apply variable neighborhood search (VNS) to the TSP flying sidekick problem, Gao applies Ant Colony Optimization (ACO) to the TSP problem [18], Huang et al. [19] applies a niching memetic algorithm (MA) to a multi-solution TSP problem, Ali, Essam and Kasmarik [20] apply a differential evolution to a discrete TSP problem, Khan and Maiti [21] apply a swap sequence based on an ABC algorithm to a TSP problem and Karaboga and Gorkemli [22] solve a TSP with a Combinatorial ABC algorithm.

From the many heuristic algorithms that exist and proof that can solve the NP-Complete problem, one interesting algorithm is to be applied in this research. The algorithm is the Artificial Bee Colony (ABC) algorithm. ABC is an algorithm developed by Dervis Karaboga, inspired by how bees find the best food sources around their hives [23].

ABC algorithm shows better results than other heuristic algorithms in several studies of various NP-Complete problems. Xu et al. [24] conducted a study by modifying the ABC algorithm to improve solution convergence. The algorithm developed is applied to the robot path planning problem. The results of this algorithm show its superiority by being superior to the comparison algorithm, namely partial swarm optimization, differential evolution, and ABC algorithm from other studies. Another study by Li et al. [25] modified the ABC algorithm to solve the multi-objective low-carbon flexible job shop scheduling problem. The results showed that improved ABC resulted in a better solution to the three comparison algorithms (MOPSO, MODE, and NSGA-II).

In the TSP problem, the ABC algorithm has also been used several times in previous studies. Choong, Wong, and Lim [26] developed the ABC algorithm with a Modified Choice Function (MCF-ABC). The modification is intended to automatically select neighborhood search heuristics in the employed and onlooker bee phases. The algorithm that has been developed is compared with several other algorithms (ABC, ACO-ABC, 2-opt ABC, TSPoptBees, bee colony optimization, hybrid discrete ABC, chained lin-kernighan, effective heuristics ACO, Quantum inspired particle swarm, and Honeybees mating optimization). The comparison studies indicate that MCF-ABC is competitive among the state-of-the-art algorithms. Another study by Venkatesh and Singh [27] also applied the ABC algorithm with a modified degree of perturbation variable to solve the generalized covering traveling salesman problem. This algorithm also shows better results than the two comparison algorithms (MA and VNS).

This research solves the TSP problem by optimizing airplane travel routes for travel. The ABC algorithm (which has good potential based on previous research) was chosen to be implemented in a hyper-heuristic framework. The case study used comes from the new dataset developed by Kiwi in a traveling salesman challenge 2.0 (TSC 2.0) competition. The results of this study compared with the GA, which is a popular algorithm in combinatorial problems and has been proven to solve TSP problems [5], [16], [28], [29].

II. Materials and Method

This section describes the process carried out in this research. The process is described in several sub-chapters where sub-chapter a discusses the dataset TSC 2.0, sub-chapter b discusses preprocessing and initial solution, sub-chapter c discusses the proposed ABC algorithm, sub-chapter d discusses experimental parameters, and sub-chapter e discusses comparison of algorithms.

A. Dataset Traveling Salesman Challenge 2.0. (TSC 2.0)

Traveling Salesman Challenge 2.0. (TSC 2.0) [30] is a competition held by the online travel company Kiwi which raises the Traveling Salesman Problem in the field of tours. The problem in TSC 2.0 aims to find an airplane route with the cheapest cost to visit several predetermined areas where one or more cities are in that area. In TSC 2.0 given 14 datasets were derived from real-world data and artificial data. The amount of data also varies from small to large. Details of the 14 datasets can be seen in table 1.

In this problem, there are also some hard constraints, such as:

- Departure city has been determined in advance
- Each area must be visited exactly once.
- Travel to the next area must continue from the city that was visited the previous day.
- Switching between areas must be done exactly 1 time every day.
- The journey must end in the same area as the departure city area.

TABLE I
DATASET TSC 2.0

Dataset	Type	Size	Number of Areas	Number of Cities	Number of days
Dataset 1	Artificial	Small	10	10	10
Dataset 2	Artificial	Small	10	15	10
Dataset 3	Artificial	Small	13	38	13
Dataset 4	Artificial	Medium	40	99	40
Dataset 5	Artificial	Medium	46	138	46
Dataset 6	Artificial	Medium	96	192	96
Dataset 7	Artificial	Large	150	300	150
Dataset 8	Artificial	Large	200	300	200
Dataset 9	Artificial	Large	250	250	250
Dataset 10	Artificial	Large	300	300	300
Dataset 11	Real	Large	150	200	150
Dataset 12	Real	Large	200	250	200
Dataset 13	Real	Large	250	275	250
Dataset 14	Real	Large	300	300	300

The main difference from classic TSP is in the presence of the terms area and city. In one solution, it is mandatory to visit each area once. However, if an area has several cities, we only need to visit one city and are free to choose any city so that the combinations that appear in this case study become more numerous.

The dataset in TSC 2.0 has the format as shown in Figure 1. The first line in each dataset contains information on the number of areas and the initial city of departure. The next few lines will contain the area name information as much as the number of areas that have been submitted in the first line. In each area it is possible to have one or more cities. The list of cities contained in an area is informed right on the line below the area name. The next line to the last line contains available flight schedule information. Each row contains information on the city of departure, city of arrival, day, and cost of the flight. On the day information, if the content is 0 then the flight is available every day.

```
10 AB0
Zona_0
AB0
Zona_1
AB1
Zona_2
AB2
Zona_3
AB3
Zona_4
AB4
Zona_5
AB5
Zona_6
AB6
Zona_7
AB7
Zona_8
AB8
Zona_9
AB9
AB0 AB1 1 1
AB0 AB2 1 3619
AB0 AB3 1 4618
AB0 AB4 1 2127
AB1 AB0 1 5639
AB1 AB2 1 4217
AB1 AB3 1 1305
AB1 AB4 1 2484
```

Fig. 1 Example dataset TSC 2.0

B. Preprocessing Data and Initial Solution

The first step in initializing the initial solution is to ensure that the city visited every day has a flight to another city the next day. The initial solution formation stage aims to prepare materials for optimization at the next stage. However, there is a problem where the initial solution cannot be made only by using random elements to choose the cities visited daily. This happens because in this case study, several combinations of cities and days lead to dead ends. This happens because in that combination it is not possible to travel the next day from that city to another city. Therefore, it is necessary to pre-process the data first

The data preprocessing stage is carried out by mapping the city into a graph on a two-dimensional array where the y-axis is a list of cities that can be visited, and the x-axis is the day of travel. Checking will be done from the first day to the last day and from the last day to the first day. If a city is not connected to another city on the next day or the previous day, then that city will be deleted. For example, in Figure 2 is an illustration of the initial conditions of mapping the city in the graph. It can be seen on the second day that city 5 is not connected to other cities on the second day. So, city 5 will be removed from the list of cities that can be visited on the second day. The impact of this deletion causes city 4 on the first day not to connect with other cities on the second day, so city 4 on the first day will also be deleted. An illustration of the results after the checking process can be seen in Figure 3. The results of data preprocessing do not guarantee that all combinations of cities and days that still exist have flights to other cities the next day. But this process reduces the chances of finding a combination that results in a dead-end

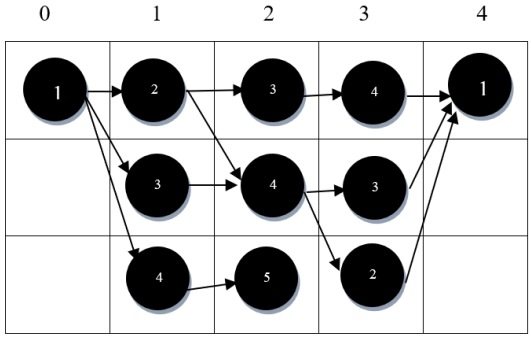

Fig. 2 Illustration of City Mapping in Graph

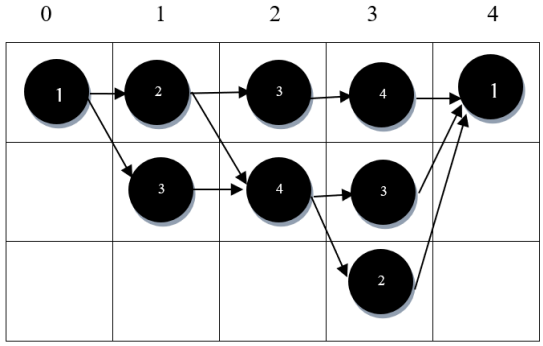

Fig. 3 Illustration of City Check Results

After preprocessing the data, the initial solution is formed. The formation of the initial solution is done by randomly selecting areas and cities. This will be done continuously until all areas have been selected and included in the initial solution. The last step is checking to ensure the resulting solution is feasible in accordance with the existing hard constraints. If it turns out that the solution is not feasible, then the solution will be deleted, and the initial solution formation stage is started again from the beginning.

C. ABC Algorithm

ABC is an algorithm developed by Karaboga [31] which is inspired by the way bees find the best food sources around their hives. In the ABC algorithm, the bee colony will be divided into three parts consisting of employed bees, onlooker bees, and scout bees [32]. The number of employed bees and onlooker bees will be equal to the number of food sources that were initiated at the beginning [33]. The way the ABC algorithm works starts with initializing the amount of nectar in all food sources carried out by the scout bee. The scout bee will provide information to the employed bee and each employed bee will go to the food source [21]. Employed bees will store the amount of nectar at the destination food source in its memory. Employed bees will look for other food sources in the vicinity. If it finds a better food source, the employed

bee will replace the contents of its memory with the amount of nectar and the location of the new food source [34]. After that, the employed bee will share information with the onlooker bee through a dance regarding the quality of the nectar found. Onlooker bee will choose some of the best food sources using probability calculations as in equation 8 [35].

$$p_i = \frac{fit_i}{\sum_{n=1}^{SN} fit_n} \quad (8)$$

Pi is the probability of the i food source obtained by dividing the value of the i food source (fit_i) by the sum of all food sources (fit_n). Some of the best food sources will be re-searched around food sources to find better food sources. If the results of one of the food sources produced at the employed bee and onlooker bee stages do not increase in several searches, then the scout bee will carry out its duties by looking for new food sources randomly and the employed bee and onlooker bee will repeat the steps from the beginning [26].

In the application of the TSC 2.0 problem, the ABC algorithm uses the low-level heuristic (LLH) as the move operator. Several LLH operators proposed in this study will be selected, consisting of swap 2 cities, swap 3 cities, swap 4 cities, and a combination of swap 3 cities and 4 cities. The selection will be done by trying on 3 representative datasets where dataset 6 represents a small dataset, dataset 11 represents a medium dataset and dataset 14 represents a large dataset. The way a 2-city swap works is by choosing from two cities from the initial solution that has been generated and then swapping the positions of the two cities. In a 3-city swap, the first step is to carry out the same process as in a 2-city swap. After that, swap 2 cities again where 1 city is chosen randomly from the 2 cities selected at the beginning, and another city is chosen randomly outside the two cities selected at the beginning. So, the city that is exchanged becomes 3 cities. In a 4-city swap, it is the same as a 3-city swap, but only the number of cities increases to 4. In a combination of 3-city and 4-city swap, random selection is made to determine the operator to be run in each iteration.

The application of this LLH will be carried out in the solution change process carried out in the ABC algorithm, namely at the employed bee and onlooker bee stages which are the stages for solution exploitation, and at the scout bee stage which is the stage for solution exploration. Figure 4 shows the final result of the algorithm used in this study. The ABC algorithm will be applied 10 times on each dataset to get average results in each dataset.

D. Parameter Experiment

Two parameters need to be adjusted to obtain better results. These parameters are the NP parameter which is the bee population, and the limit parameter, which functions to set the stage when the exploration stage is carried out and the exploitation stage. Experiments will be carried out by trying random parameter values and decreasing or adding extreme values to find the optimal value estimate. After getting the estimated optimal value, a trial is carried out by adding and subtracting the parameter value by 50% to ensure that the value used is the closest to the best value.

E. Comparison of Algorithms

Because very few studies use the same dataset, to see the performance of the results of this study, GA will be used as a comparison algorithm. The application of GA will be carried out the same as with ABC using the same initial solution and the same parameter experimental method. Experiments will also be carried out 10 times on each dataset with the same amount of time as the ABC algorithm.

```
1: procedure Artificial Bee Colony
2:    NP ← initial bee colony
3:    Limit ← initial limit
4:    FoodSource ← initial solution
5:    While ~Stop Condition() Do
6:       Employed Bee phase
7:       For each employed bee
8:          Do Move Strategy in FoodSouce
9:          Calculate value fit
10:         Apply greedy selection mechanism
11:      End
12:      Calculate the probability value pi for the solution
13:      Onlooker Bee phase
14:         Chooses a food source depending on pi
15:         Do Move Strategy in FoodSouce
16:         Apply greedy selection mechanism
17:      End
18:      Scout Bee Phase
19:      If there is an employed bee becomes scout Then
20:      Do Move Strategy in FoodSouce
21:      Memorize the best solution achieved so far
22:   End
```

Fig. 4 Pseudocode Artificial Bee Colony Algorithm

III. Result and Discussion

All experiments ran on an Intel's Core i7 2.6 GHz computer with 16 GB of RAM under the Windows 10 64-bit operating system. The programming language used was java with idea NetBeans 8.2.

A. Preprocessing Data and Initial Solution

The first step to solving a TSP problem is to find an initial solution. This process is carried out by preprocessing the data by removing the possibility of a city that can be visited on certain days, which will lead to a dead-end route. The next stage is forming an initial solution by choosing a city randomly. This stage succeeded in finding 14 datasets with feasible solutions. Preprocessing data plays an important role in this stage. Without preprocessing the data, only two datasets (datasets 1 and 3) can find a feasible solution.

B. Parameter Experiment

The first experiment was to test the limit parameters. Datasets number six chosen to represent small and medium artificial dataset, dataset number 11 chosen to represent large artificial dataset, and dataset number 14 chosen to represent real-world datasets. Parameter testing begins by finding the approximate range of optimal parameter values. The result is that there are three possible values, namely 2500, 5000, and 7500. These three values are applied to datasets 6, 11, 14 11

times. Table 2 shows the average results of the three limit parameter values. These results show that the limit parameter with a value of 5000 can produce the most stable results in small, medium, and large datasets. Therefore, the limit parameter value of 5000.

TABLE II
TEST RESULTS LIMIT PARAMETER

Limit Value	Dataset 6	Dataset 11	Dataset 14
2500	3668.1	61681.8	151380
5000	3723.9	62113.8	150909
7500	3777.1	62491.1	152161

The next experiment was carried out on the NP parameters. With the same method applied to the limit parameters, values of 4, 8, and 12 were obtained to be tested on datasets 6, 11, and 14 11 times. This experiment results in the parameter value 8 being the optimal value. However, the difference obtained is only around 1-2% and is not comparable to the increase in running time, almost two times. Therefore, the NP parameter value to be used is 4. The test results can be seen in Table 3.

TABLE III
TEST RESULTS NP PARAMETER

NP Value	Dataset 6	Dataset 11	Dataset 14
4	3668,1	61681,8	151380
8	3610,8	60957,6	149571,4
12	3623,1	61065,6	148643,6

C. *Move Strategy*

In this section, the movement strategy is selected by applying several combinations of LLH that can produce the most optimal solution. Four types of operator swaps were applied to datasets 6, 11, and 14 for 11 repetitions. The results show that swap 4 city operators produce the best solution on dataset number six. In datasets 11 and 14, the combination of swap operators 3 and 4 cities produced the best results. The 4-city swap operator will produce a more diverse solution. However, the 4-city swap operator allows for a better solution when just swapping 2 or 3 cities. So the combination method between swap 3 and 4 cities produces the best results.

TABLE IV
TEST RESULT OF MOVE STRATEGY

LLH	Dataset 6	Dataset 11	Dataset 14
Swap 2-City	3668,1	61681,8	151380
Swap 3-City	3429,8	61242,6	148982
Swap 4-City	3351,5	61196,5	148956,7
Swap 3&4-City	3420	60453.7	148304.6

D. *Result of Implementation ABC Algorithm*

After finding the parameter values and move strategy that can produce optimal results, then the ABC algorithm is applied to 14 datasets with 500.000.000 iteration for 11 repetitions. The results can be seen in Table 5 and Figure 5. Based on the comparison with the initial solution, the ABC algorithm can optimize travel costs with an average of 54.6% smaller than the results produced by the initial solution. The ABC algorithm only produces the same solution in dataset 2 because the results in the initial solution are already the optimal results.

These results can be interpreted based on artificial datasets and real-world datasets. In the artificial dataset, the ABC algorithm can reduce costs by an average of 62.96%. If dataset number two is omitted from the calculation because the initial solution is already the optimal solution, the ABC algorithm can reduce costs by an average of 69.95%. The difference in size from small to large datasets does not significantly differ in the optimization results. These results show that the ABC algorithm can optimize the solution well on artificial datasets.

TABLE V
RESULT ABC ALGORITHM COMPARED TO INITIAL SOLUTION

Dataset	Initial Solution	ABC	Cost Reduction (%)
1	8609	1396	83,78%
2	1498	1498	0,00%
3	17818	9032,2	49,31%
4	52623	17313,5	67,10%
5	5157	911,2	82,33%
6	12625	3420	72,91%
7	89234	33238,1	62,75%
8	17434	7518,1	56,88%
9	290733	83534,2	71,27%
10	374530	62852,8	83,22%
11	94610	61681,8	34,80%
12	147314	85024,8	42,28%
13	184021	142147	22,76%
14	230839	148304,6	35,75%

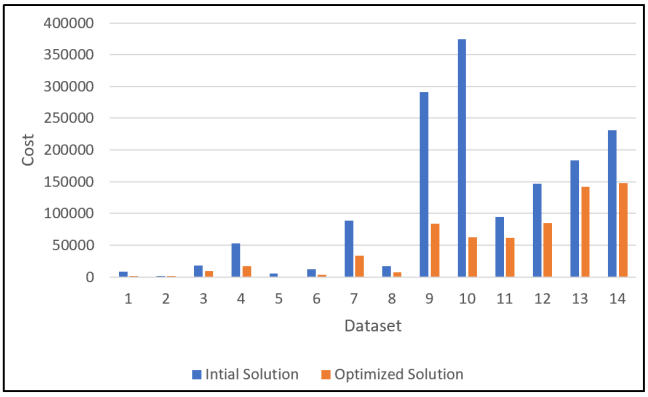

Fig. 5 Comparison of initial solution and optimized solution

Different results exist in real-world datasets. The ABC algorithm can only optimize costs with an average of 33.9%. Based on the number of areas, cities, and days, there is no significant difference between real-world and large artificial datasets. The ABC algorithm can still optimize costs in large artificial datasets with cost reductions above 50%. So it can be concluded that unknown factors distinguish between artificial and real-world datasets in the TSC 2.0 problem.

E. *Result Compared to GA*

GA is run to 14 datasets with 500,000,000 iterations for 11 repetitions. The results of the comparison of the ABC and GA algorithms can be seen in Table 6 and Figure 6. From these results, it can be seen that the ABC algorithm outperform in 13 data sets from 14 data sets. Only one dataset obtained the same result, namely dataset 2, because indeed from the initial results the solution was already the optimal solution and indeed there was only one available solution. In percentage terms, the ABC algorithm produces more cost-effectiveness than GA with an average of 26% savings.

When compared to the problem's size, the ABC algorithm optimizes the solution with an average of 32% on small and medium datasets and 22% on large datasets. Compared to the type of problem, the ABC algorithm optimizes the solution with an average of 27% on the large artificial dataset and 17% on the real-world dataset. From all aspects, the ABC algorithm outperforms GA.

Another difference is in the variation of the resulting solution. The ABC algorithm produces solutions with smaller solution variations than the GA algorithm on the entire dataset. Figures 7, 8, and 9 compare the two algorithms from the boxplot graph. Smaller solution variation tends to find results similar to the average found in this study if the algorithm is re-run on the same problem. Meanwhile, the larger solution variation will increase the possibility of producing a solution with a rather large difference from the average found in this study.

TABLE VI
COMPARISON BETWEEN ABC ALGORITHM AND GA

Dataset	ABC	GA	Cost Reduction (%)
1	1396	1986,4	30%
2	1498	1498	0%
3	9032,2	10874,4	17%
4	17313,5	34243	49%
5	911,2	3366,5	73%
6	3420	4332,4	21%
7	33238,1	38000,9	13%
8	7518,1	8674,2	13%
9	83534,2	126242	34%
10	62852,8	120034,4	48%
11	61681,8	67614,4	9%
12	85024,8	97529,4	13%
13	142147	170819,8	17%
14	148304,6	209126,3	29%

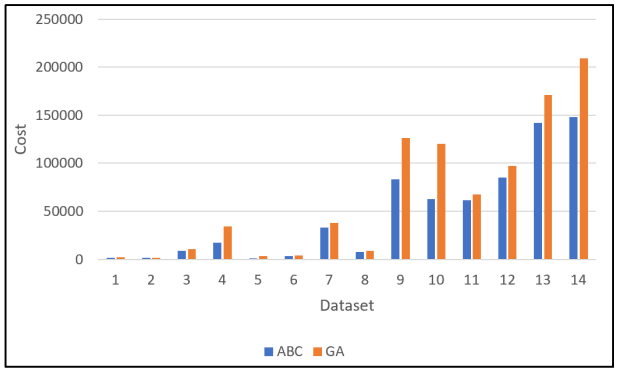

Fig. 6 Comparison of ABC and GA

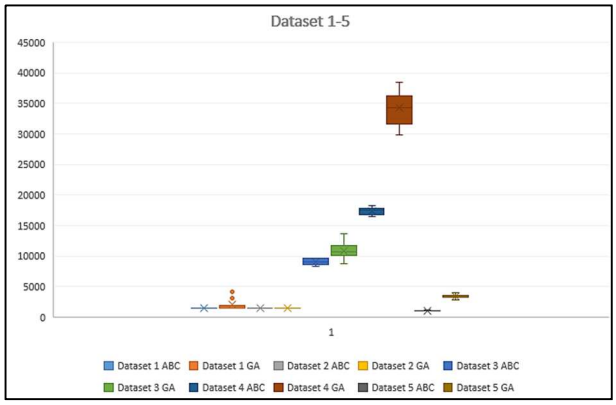

Fig. 7 Boxplot dataset 1-5

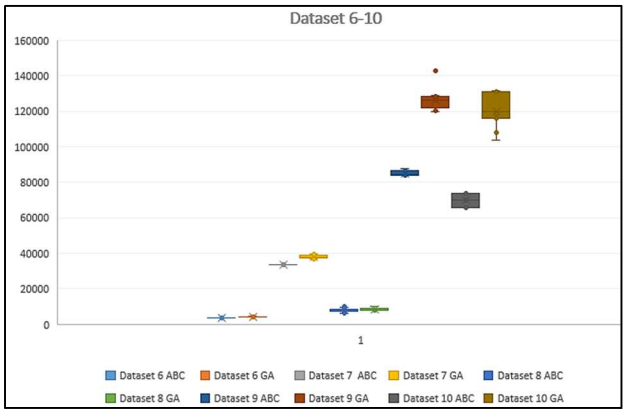

Fig. 8 Boxplot dataset 6-10

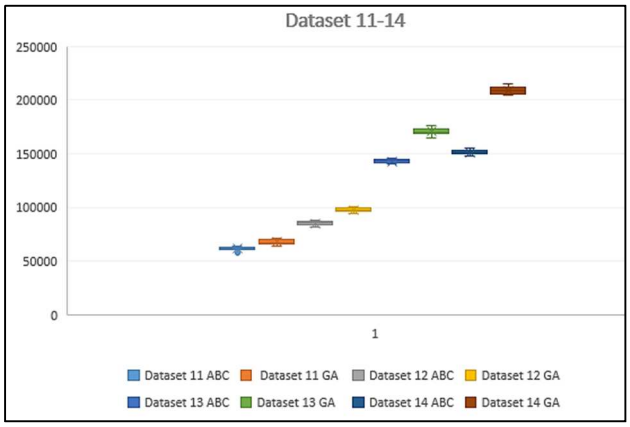

Fig. 9 Boxplot dataset 11-14

IV. CONCLUSION

TSP is a combinatoric optimization problem that belongs to the NP-Hard category. This research completes a case study of TSP problems published by Kiwi. This problem contains how to find the cheapest air transportation travel costs to be able to visit each area exactly once. ABC algorithm is proposed to find the shortest route. Several combinations of operator swaps were applied to this problem, ranging from swaps between two cities, swaps between three cities, swaps between 4-cities, and swap combinations between 3 and 4-cities. The experimental results show that the overall swap combination between 3 and 4 produces the best results. In addition, the NP and limit parameters were tested. The result is that the NP parameter value is 4 and the limit parameter value of 5000 is the optimal value. After finding the optimal parameter values and swap operator, the ABC algorithm is executed on each dataset with a total of 500,000,000 iterations. The result is that this algorithm can reduce travel costs by an average of 54.6%. This research applies the same problem to the GA as a comparison algorithm. The result is that the ABC algorithm can find solutions with cheaper travel costs by an average of 26%. This approach may be improved in the future by trying different experiments and swap operator combinations. In addition, this approach can also be tested on other TSP problems with more real-world dataset.

REFERENCES

[1] M. A. H. Akhand, S. I. Ayon, S. A. Shahriyar, N. Siddique, and H. Adeli, "Discrete Spider Monkey Optimization for Travelling Salesman Problem," *Appl. Soft Comput. J.*, vol. 86, 2020, doi: 10.1016/j.asoc.2019.105887.

[2] G. Campuzano, C. Obreque, and M. M. Aguayo, "Accelerating the Miller–Tucker–Zemlin model for the asymmetric traveling salesman problem," *Expert Syst. Appl.*, vol. 148, 2020, doi: 10.1016/j.eswa.2020.113229.

[3] E. Baş and E. Ülker, "Dıscrete socıal spıder algorıthm for the travelıng salesman problem," *Artif. Intell. Rev.*, vol. 54, no. 2, 2021, doi: 10.1007/s10462-020-09869-8.

[4] Komarudin and S. F. Parhusip, "Composite algorithm based on Clarke – Wright and local search for the traveling salesman problem," 2019. doi: 10.1145/3364335.3364388.

[5] M. A. Al-Furhud and Z. Hussain, "Genetic Algorithms for the Multiple Travelling Salesman Problem," *Int. J. Adv. Comput. Sci. Appl.*, vol. 11, no. 7, 2020, doi: 10.14569/IJACSA.2020.0110768.

[6] A. C. Cinar, S. Korkmaz, and M. S. Kiran, "A discrete tree-seed algorithm for solving symmetric traveling salesman problem," *Eng. Sci. Technol. an Int. J.*, vol. 23, no. 4, pp. 879–890, Aug. 2020, doi: 10.1016/j.jestch.2019.11.005.

[7] J. Kaur and A. Pal, "An analysis of different metaheuristic approaches for solving travelling salesman problems," *Adv. Math. Sci. J.*, vol. 9, no. 8, 2020, doi: 10.37418/amsj.9.8.29.

[8] M. Mosayebi, M. Sodhi, and T. A. Wettergren, "The Traveling Salesman Problem with Job-times (TSPJ)," *Comput. Oper. Res.*, vol. 129, 2021, doi: 10.1016/j.cor.2021.105226.

[9] N. Rokbani *et al.*, "Bi-heuristic ant colony optimization-based approaches for traveling salesman problem," *Soft Comput.*, vol. 25, no. 5, 2021, doi: 10.1007/s00500-020-05406-5.

[10] M. A. Tawhid and P. Savsani, "Discrete Sine-Cosine Algorithm (DSCA) with Local Search for Solving Traveling Salesman Problem," *Arab. J. Sci. Eng.*, vol. 44, no. 4, 2019, doi: 10.1007/s13369-018-3617-0.

[11] Z. Daoqing and J. Mingyan, "Parallel discrete lion swarm optimization algorithm for solving traveling salesman problem," *J. Syst. Eng. Electron.*, vol. 31, no. 4, 2020, doi: 10.23919/JSEE.2020.000050.

[12] S. K. R. Kanna, K. Sivakumar, and N. Lingaraj, "Development of Deer Hunting linked Earthworm Optimization Algorithm for solving large scale Traveling Salesman Problem," *Knowledge-Based Syst.*, vol. 227, 2021, doi: 10.1016/j.knosys.2021.107199.

[13] I. G. A. Premananda and A. Muklason, "Complex University Timetabling Using Iterative Forward Search Algorithm and Great Deluge Algorithm," *Khazanah Inform. J. Ilmu Komput. dan Inform.*, vol. 7, no. 2, 2021.

[14] C. Jiang, Z. Wan, and Z. Peng, "A new efficient hybrid algorithm for large scale multiple traveling salesman problems," *Expert Syst. Appl.*, vol. 139, 2020, doi: 10.1016/j.eswa.2019.112867.

[15] R. S. de Moraes and E. P. de Freitas, "Experimental analysis of heuristic solutions for the moving target traveling salesman problem applied to a moving targets monitoring system," *Expert Syst. Appl.*, vol. 136, 2019, doi: 10.1016/j.eswa.2019.04.023.

[16] Q. M. Ha, Y. Deville, Q. D. Pham, and M. H. Hà, "A hybrid genetic algorithm for the traveling salesman problem with drone," *J. Heuristics*, vol. 26, no. 2, 2020, doi: 10.1007/s10732-019-09431-y.

[17] J. C. de Freitas and P. H. V. Penna, "A variable neighborhood search for flying sidekick traveling salesman problem," *Int. Trans. Oper. Res.*, vol. 27, no. 1, pp. 267–290, 2020, doi: 10.1111/itor.12671.

[18] W. Gao, "New ant colony optimization algorithm for the traveling salesman problem," *Int. J. Comput. Intell. Syst.*, vol. 13, no. 1, 2020, doi: 10.2991/ijcis.d.200117.001.

[19] T. Huang, Y. J. Gong, S. Kwong, H. Wang, and J. Zhang, "A Niching Memetic Algorithm for Multi-Solution Traveling Salesman Problem," *IEEE Trans. Evol. Comput.*, vol. 24, no. 3, 2020, doi: 10.1109/TEVC.2019.2936440.

[20] I. M. Ali, D. Essam, and K. Kasmarik, "A novel design of differential evolution for solving discrete traveling salesman problems," *Swarm Evol. Comput.*, vol. 52, 2020, doi: 10.1016/j.swevo.2019.100607.

[21] I. Khan and M. K. Maiti, "A swap sequence based Artificial Bee Colony algorithm for Traveling Salesman Problem," *Swarm Evol. Comput.*, vol. 44, 2019, doi: 10.1016/j.swevo.2018.05.006.

[22] D. Karaboga and B. Gorkemli, "Solving Traveling Salesman Problem by Using Combinatorial Artificial Bee Colony Algorithms," *Int. J. Artif. Intell. Tools*, vol. 28, no. 1, 2019, doi: 10.1142/S0218213019500040.

[23] M. R. Batchanaboyina and N. R. Devarakonda, "Handling optimization problem, and the scope of varied artificial bee colony (ABC) algorithms: A contemporary research," *Int. J. Innov. Technol. Explor. Eng.*, vol. 8, no. 6 Special Issue 4, 2019, doi: 10.35940/ijitee.F1125.0486S419.

[24] F. Xu *et al.*, "A new global best guided artificial bee colony algorithm with application in robot path planning," *Appl. Soft Comput. J.*, vol. 88, 2020, doi: 10.1016/j.asoc.2019.106037.

[25] Y. Li, W. Huang, R. Wu, and K. Guo, "An improved artificial bee colony algorithm for solving multi-objective low-carbon flexible job shop scheduling problem," *Appl. Soft Comput. J.*, vol. 95, 2020, doi: 10.1016/j.asoc.2020.106544.

[26] S. S. Choong, L. P. Wong, and C. P. Lim, "An artificial bee colony algorithm with a Modified Choice Function for the traveling salesman problem," *Swarm Evol. Comput.*, vol. 44, 2019, doi: 10.1016/j.swevo.2018.08.004.

[27] V. Pandiri and A. Singh, "An artificial bee colony algorithm with variable degree of perturbation for the generalized covering traveling salesman problem," *Appl. Soft Comput. J.*, vol. 78, 2019, doi: 10.1016/j.asoc.2019.03.001.

[28] A. M. H. Al-Ibrahim, "Solving Travelling Salesman Problem (TSP) by Hybrid Genetic Algorithm (HGA)," *Int. J. Adv. Comput. Sci. Appl.*, vol. 11, no. 6, 2020, doi: 10.14569/IJACSA.2020.0110649.

[29] A. Riazi, "Genetic algorithm and a double-chromosome implementation to the traveling salesman problem," *SN Appl. Sci.*, vol. 1, no. 11, 2019, doi: 10.1007/s42452-019-1469-1.

[30] Kiwi, "Travelling Salesman Challenge 2.0," 2019. https://travellingsalesman.kiwi.com.

[31] K. Hussain, M. N. Mohd Salleh, S. Cheng, Y. Shi, and R. Naseem, "Artificial bee colony algorithm: A component-wise analysis using diversity measurement," *J. King Saud Univ. - Comput. Inf. Sci.*, vol. 32, no. 7, 2020, doi: 10.1016/j.jksuci.2018.09.017.

[32] H. C. Tsai, "Artificial bee colony directive for continuous optimization," *Appl. Soft Comput. J.*, vol. 87, 2020, doi: 10.1016/j.asoc.2019.105982.

[33] M. A. Awadallah, M. A. Al-Betar, A. L. Bolaji, I. A. Doush, A. I. Hammouri, and M. Mafarja, "Island artificial bee colony for global optimization," *Soft Comput.*, vol. 24, no. 17, 2020, doi: 10.1007/s00500-020-04760-8.

[34] W. li Xiang, Y. zhen Li, R. chun He, and M. qing An, "Artificial bee colony algorithm with a pure crossover operation for binary optimization," *Comput. Ind. Eng.*, vol. 152, 2021, doi: 10.1016/j.cie.2020.107011.

[35] Y. Deng, H. Xu, and J. Wu, "Optimization of blockchain investment portfolio under artificial bee colony algorithm," *J. Comput. Appl. Math.*, vol. 385, 2021, doi: 10.1016/j.cam.2020.113199.