

# Hybrid iterated local search algorithm for optimization route of airplane travel plans

Ahmad Muklason, I Gusti Agung Premananda

Department of Information Systems, Institut Teknologi Sepuluh Nopember, Surabaya, Indonesia

---

## Article Info

### Article history:

Received Dec 6, 2022

Revised Jan 20, 2023

Accepted Feb 4, 2023

---

### Keywords:

Hyper-heuristic

Iterated local search

Simulated annealing

Traveling Salesman Challenge 2.0

Traveling salesman problem

---

## ABSTRACT

The traveling salesman problem (TSP) is a very popular combinatorics problem. This problem has been widely applied to various real problems. The TSP problem has been classified as a Non-deterministic Polynomial Hard (NP-Hard), so a non-deterministic algorithm is needed to solve this problem. However, a non-deterministic algorithm can only produce a fairly good solution but does not guarantee an optimal solution. Therefore, there are still opportunities to develop new algorithms with better optimization results. This research develops a new algorithm by hybridizing three local search algorithms, namely, iterated local search (ILS) with simulated annealing (SA) and hill climbing (HC), to get a better optimization result. This algorithm aimed to solve TSP problems in the transportation sector, using a case study from the Traveling Salesman Challenge 2.0 (TSC 2.0). The test results show that the developed algorithm can optimize better by 15.7% on average and 11.4% based on the best results compared to previous studies using the Tabu-SA algorithm.

*This is an open access article under the [CC BY-SA](https://creativecommons.org/licenses/by-sa/4.0/) license.*



---

## Corresponding Author:

Ahmad Muklason

Department of Information Systems, Institut Teknologi Sepuluh Nopember

Jl. Raya ITS, ITS Sukolilo Campus, Surabaya, 60111, East Java, Indonesia

Email: mukhlason@is.its.ac.id

---

## 1. INTRODUCTION

Traveling salesman problem (TSP) is a classic problem that is still popularly researched in combinatorics optimization [1]. This problem discusses arranging the order of places to be visited so that all places are only visited once [2]. In addition, the place where the trip ends must be the same as where the trip started [3]. This problem generally aims to obtain travel routes with the least distance or the cheapest costs [1], [4]. The TSP problem is becoming popular because it has many applications in real problems. Some examples of real-world problems that apply TSP include transportation, delivery, search, rescue, precision agriculture, monitoring, and surveillance [5].

TSP problems have been classified as non-deterministic polynomial hard (NP-Hard) problems. Consequently, this problem can only solve using a non-deterministic algorithm [6]. The non-deterministic algorithm works by modifying the solution and will sort out whether the modified results are used in the next iteration. In TSP problems, modifications are generally carried out by swapping between two or more cities. At the same time, the selection of modified solutions is carried out according to the algorithm used with the primary goal of reducing the distance or cost of the resulting solutions. However, non-deterministic algorithms can only optimize the solution to produce a good enough solution but cannot guarantee that the solution is optimal [7]. Therefore, the TSP problem is still an open problem.

Various studies have tried to develop algorithms to solve TSP problems. In the last ten years, the hybrid method has become a popular method. This popularity is due to the advantages of the hybridization

method, which can take advantage of two different algorithms and make a better solution. Several hybrid methods have been developed for TSP problems, such as hybridization between genetic algorithm (GA) and local search [8], ant colony optimization (ACO) and particle swarm optimization (PSO) [9], and ACO and simulated annealing (SA) [10].

Of the various kinds of algorithms, three local search algorithms show promising solutions in solving combinatoric optimization problems, namely the iterated local search (ILS), simulated annealing (SA) and hill climbing (HC) algorithms. The ILS algorithm has been proven to produce promising solutions for various kinds of optimization problems, such as timetabling [11]–[13], vehicle routing problems [14]–[16], and TSP [17], [18]. Meanwhile, the SA algorithm also shows promising results, such as in the fields of bin packing [19], timetabling [20]–[22], and TSP [23]–[25]. Meanwhile, the HC algorithm is widely used in the development of hybridization algorithms to improve the final results of the optimization process [26].

This research develops a new algorithm by hybridizing three local search algorithms: ILS, SA, and HC. The algorithm develops using the hyper-heuristic method. This algorithm is intended to solve one of the TSP problems, namely the Traveling Salesman Challenge 2.0 (TSC 2.0) case study. The results of the algorithm development show out-performing results from Ahmad *et al.* [25], which used a hybridization algorithm between simulated annealing and Tabu search (TS).

## 2. RELATED WORK

The development of hybrid algorithms to solve TSP problems has been widely developed. In the last five years, several algorithms have been developed aimed at various kinds of TSP problems. Most of the development is done by combining fellow local search algorithms or combining population algorithms with local search algorithms. Dong *et al.* [27] developed a hybridization between the Wolf pack search algorithm and the local search algorithm developed by other researchers. This algorithm is intended to solve TSP problems on 12 instances on the TSPLIB benchmark. The research results show that the developed algorithm is superior to the seven previous studies.

Ha *et al.* [28] a hybrid algorithm was developed by combining a GA with local search algorithms (hill-climbing and first-improvement local search) to solve TSP problems with drones. The research also introduced a new crossover operator. The algorithm was found to be superior to several previous studies and discovered new best-known solutions. Another hybrid algorithm was developed in 2020 by combining the TS and SA algorithms [25]. The SA algorithm served as the primary method for finding solutions while the TS algorithm was used to increase diversity for low-level heuristic (LLH). This hybrid algorithm was tested on TSC 2.0 problems and was found to produce better results than a single local search algorithm, such as the great deluge algorithm.

Deng *et al.* [29] created a hybrid algorithm by combining the cellular genetic algorithm (CGA) with the simulated annealing (SA) algorithm. The SA algorithm was employed to optimize solutions that were modified through crossover and mutation operators. This hybrid algorithm was specifically designed to address TSP problems using the TSPLIB benchmark. The results of this study revealed that the hybrid algorithm performed better than either CGA or SA alone.

## 3. CASE STUDY

TSC 2.0 is a competition held by Kiwi, a company engaged in traveling. This competition raises the problem of arranging travel routes for travelers that want to visit several areas. This problem follows the form of the TSP problem, where each area will be visited exactly once, and the end of the trip, must return to where the trip first started. In this problem, each area can have several cities. In one area visited, the traveler can only choose to visit one city. This problem aims to minimize the cost of plane tickets that must be incurred to make this trip.

TSC 2.0 provides fourteen datasets consisting of 10 artificial datasets and four real-world datasets. Based on the number of areas and cities, the fourteen datasets can be classified into three: small datasets consisting of datasets 1-3, medium datasets consisting of datasets 4-6, and large datasets consisting of datasets 7-14. Table 1 (as shown in appendix) shows the description of each dataset.

## 4. METHOD

### 4.1. Pre-processing data

In the TSC 2.0 problem, not all cities have flights to other cities. So, in forming the initial solution, it is possible to find cities that do not have flights to cities in the next area that has yet to be visited. At this stage, each area will be arranged in a two-dimensional matrix where the x-axis represents the day, and the y-axis represents the city. Each city on each day will be checked to see whether there are flights to other cities

the following day. If a city is found that has no flights to another city the next day, that city will be deleted. Figure 1 shows how to illustrate this data preprocessing.

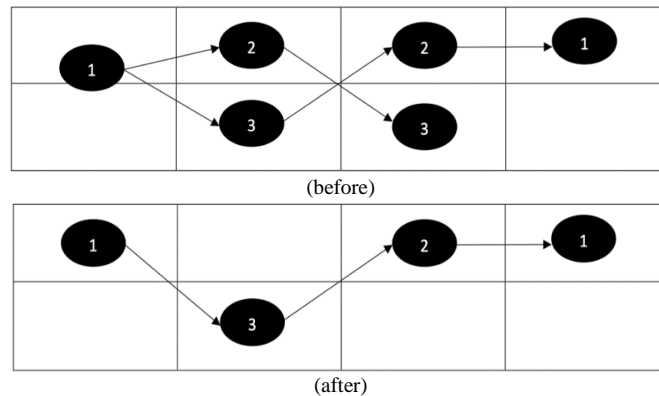


Figure 1. Illustration of pre-processing data

#### 4.2. Formation initial solution

The initial solution is generated by executing the steps in Figure 2. The first step is initializing variables such as *indexDay*, *initialSolution*, *listCity*, and *indexStuck*. The algorithm repeatedly runs through certain stages until a feasible solution is found. Each iteration randomly chooses areas and cities for *indexDay* and then checks if there are any flights from that city to another city on the next day. If there are no flights, that area and city are not used, and the *indexStuck* variable is incremented. On the other hand, if the area and city are used, the *indexStuck* variable is reset to 0, and the process continues for the next day. If the *indexStuck* value reaches 500, the algorithm starts over to avoid getting stuck in a local optimal solution.

```

Function InitialSolution (MatrixCityDay):
  indexDay=0;
  initialSolution={};
  listCity=preprocessingData(MatrixCityDay);
  indexStuck=0;
  while !solutionFeasible(initialSolution) do
    selectCity(indexDay);
    if checkNextCity(initialSolution[indexDay]) then
      indexStuck=0;
      indexDay++;
    end
    indexStuck++;
    if indexStuck==500 then
      indexDay=0;
      initialSolution={};
    end
  end
  return initialSolution;

```

Figure 2. Pseudocode formation initial solution

#### 4.3. Development moves acceptance algorithm

This study created a hybrid move acceptance algorithm by combining ILS, SA, and HC algorithms. Figure 3 illustrates the pseudocode of the developed hybridization scheme. The ILS algorithm is the foundation of the hybrid algorithm and has two phases: a perturbation stage for modifying solutions and an improvement stage for refining solutions. The SA algorithm is used during the perturbation stage to broaden the search for solutions, while the HC algorithm is used during the improvement stage to optimize the solutions.

In the perturbation process, the SA algorithm is employed as outlined in Figure 4. The SA algorithm modifies the current solution by randomly applying LLH and then determining whether to accept the new solution by comparing it to the current one. If the new solution is better, it is used in the next iteration, but if it is worse, a probability calculation is performed using (1).

$$P = e^{-\frac{c}{t}} \quad (1)$$

where  $P$  is probability,  $e$  is exponential number,  $c$  is the difference between the current and new solutions, and  $t$  is the temperature parameter. The temperature variable is then reduced by multiplying it by a cooling rate, and these steps are repeated until the temperature falls below 1.

During the improvement stage of the process, the hill climbing (HC) algorithm is applied as outlined in Figure 5. The HC algorithm runs a set number of iterations. In every iteration, it selects a random LLH to generate a new solution. If the new solution is an improvement over the current one, it is adopted.

```

Function
OptimizationSolution (initialSolution, MaxIteration, iterationSA, iterationHC) :
  Solution=IntialSolution;
  bestSolution=Solution;
  bestFitness=calculateFitness(bestSolution);
  while !MaxIteration do
    Solution=ApplySA(Solution,iterationSA);
    Solution=ApplyHC(Solution,iterationHC);
    if calculateFitness(Solution);bestFitness then
      bestSolution=Solution;
      bestFitness=calculateFitness(bestSolution);
    end
  end
  return bestSolution;

```

Figure 3. Pseudocode ILS algorithm

```

Function SimulatedAnnealing (Solution, temperature, coolingRate) :
  fitnessSolution=calculateFitness(Solution);
  bestSolution=Solution;
  bestFitness=calculateFitness(bestSolution);
  while temperature > 1 do
    Solution*=ApplyLLH(Solution);
    fitnessSolution*=calculateFitness(Solution*)
    p=Math.exp((fitnessSolution-fitnessSolution*)/temperature);
    if fitnessSolution* < fitnessSolution or p <= random() then
      Solution=Solution*;
      fitnessSolution=fitnessSolution*;
    end
    if fitnessSolution;bestFitness then
      bestSolution=Solution;
      bestFitness=fitnessSolution;
    end
    temperature=temperature*coolingRate;
  end
  return bestSolution;

```

Figure 4. Pseudocode SA algorithm

```

Function HillClimbing (Solution, MaxIteration) :
  fitnessSolution=calculateFitness(Solution);
  bestSolution=Solution;
  bestFitness=calculateFitness(bestSolution);
  while !MaxIteration do
    Solution*=ApplyLLH(Solution);
    fitnessSolution*=calculateFitness(Solution*)
    if fitnessSolution* < fitnessSolution then
      Solution=Solution*;
      fitnessSolution=fitnessSolution*;
    end
    if fitnessSolution;bestFitness then
      bestSolution=Solution;
      bestFitness=fitnessSolution;
    end
  end
  return bestSolution;

```

Figure 5. Pseudocode HC algorithm

#### 4.4. Development low-level heuristic

In this study, the LLH used is the swap operator, which functions to exchange positions between two cities. This operator swap was also developed into two new LLH: swap 2x and swap 3x. Swap 2x will run the swap operator twice. At the same time, the 3x swap will run the swap operator three times. The three LLH swaps will be chosen randomly at each iteration. In the 2x and 3x LLH swaps, a backtracking procedure was developed, which functions to roll back once the swap operator applied to produce a new solution and be sent back at the move acceptance stage.

#### 4.5. Parameter setting

Some of the parameter variables in the developed algorithm need to be searched for optimal values so that the algorithm runs well. The first is the number of iterations. Research [25] which is used as a comparison, runs the algorithm with parameter settings equivalent to 46,051,691 iterations. This research uses parameter settings designed to get the same number of iterations to get a fair comparison. In addition, this study also performs parameter settings on the composition between the SA and HC algorithms. The five trial scenarios that were run are presented in Table 2.

**Table 2. Parameter experiment**

Simulated Annealing	Hill Climbing
4%	1%
3%	2%
2.5%	2.5%
2%	3%
1%	4%

## 5. RESULT AND DISCUSSION

### 5.1. Pre-processing data and formation initial solution

The application of the pre-processing data stage can assist the initial solution formation process. Only the first dataset can find a feasible solution without applying data pre-processing. However, after implementing the pre-processing data stages, all datasets were found to have a feasible initial solution.

### 5.2. Optimization result

At this stage, 15 trials were carried out on each dataset for each scenario. Tables 3 and 4 show the average and best results for each dataset. Overall, it was found that the composition of the algorithm SA 4% and HC 1% produced the best results. In terms of average, the average composition of SA 4% and HC 1% produced the best results in 11 datasets. Meanwhile, based on the best results, the composition of SA 4% and HC 1% produced the best results in 7 datasets. These results show that running the SA algorithm in a larger composition in ILS algorithm, can produce a better solution because in running the SA algorithm, there are two processes of exploitation and exploration, which cause the search for solutions to run convergently but not trapped in local optimal conditions. At the same time, the HC algorithm serves as a supporting algorithm to improve the solution.

**Table 3. Average result**

Dataset	SA 4 HC 1	SA 3 HC 2	SA 2.5 HC 2.5	SA 2 HC 3	SA 1 HC 4
1	1396.0	1396.0	1396.0	1396.0	1396.0
2	1498.0	1498.0	1498.0	1498.0	1498.0
3	9302.1	9237.7	9419.7	9298.5	9357.7
4	17803.2	17803.4	18038.0	17962.9	18226.1
5	1078.1	1068.9	1083.7	1126.3	1123.6
6	3799.4	3869.6	3776.7	3882.5	4160.9
7	33766.1	33985.3	34193.7	34215.1	35070.8
8	6635.3	6795.1	6874.7	6963.2	7406.5
9	124553.8	127230.7	131211.5	134366.8	143643.8
10	100420.7	102590.8	103998.2	108842.8	118035.5
11	62713.7	63456.9	63206.9	65022.2	67081.8
12	87662.7	89071.3	90590.5	90355.3	94489.4
13	156486.2	159016.3	159777.5	160538.9	163457.7
14	195538.7	200089.8	199039.0	200656.5	201838.7

**Table 4. Best result**

Dataset	SA 4 HC 1	SA 3 HC 2	SA 2.5 HC 2.5	SA 2 HC 3	SA 1 HC 4
1	1396	1396	1396	1396	1396
2	1498	1498	1498	1498	1498
3	8807	8571	8714	8888	8470
4	15741	16038	16890	16698	16821
5	933	945	849	906	1005
6	3393	3578	3461	3611	3765
7	33071	33528	33417	33589	34004
8	5383	5511	6420	5412	6541
9	120598	120814	126954	129918	132954
10	94014	90173	98050	102156	110354
11	61409	59520	60735	62168	61603
12	85555	84377	88579	87669	90061
13	153679	154569	154832	156340	160145
14	191004	196700	194813	195071	191179

### 5.3. Comparison with previous studies

To see the performance of the developed algorithm, a comparison was made with previous studies. The previous study [25] developed a hybridization algorithm from the SA and Tabu search algorithms. The

two algorithms are compared in Table 5, Figures 6 and 7. The results showed that the developed algorithm outperformed previous research by being able to optimize 15.7% better based on the average results and 11.4% better based on the best results from the entire dataset.

**Table 5. Comparison against Tabu-simulated annealing**

Dataset	Hybrid ILS		Tabu SA	
	Best	Average	Best	Average
1	1396	1396	1396	1396
2	1498	1498	1498	1498
3	8807	8571	8714	8888
4	15741	16038	16890	16698
5	933	945	849	906
6	3393	3578	3461	3611
7	33071	33528	33417	33589
8	5383	5511	6420	5412
9	120598	120814	126954	129918
10	94014	90173	98050	102156
11	61409	59520	60735	62168
12	85555	84377	88579	87669
13	153679	154569	154832	156340
14	191004	196700	194813	195071

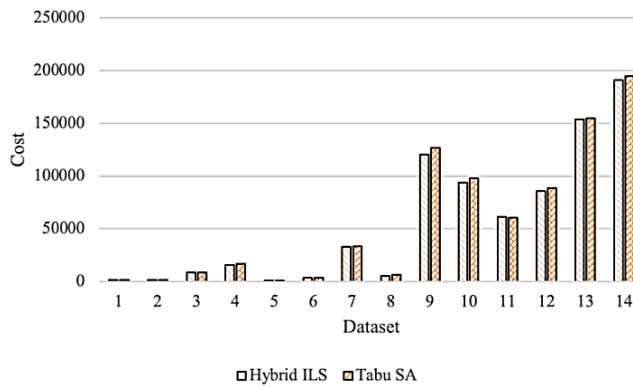


Figure 6. Comparison hybrid ILS and Tabu-SA on best result

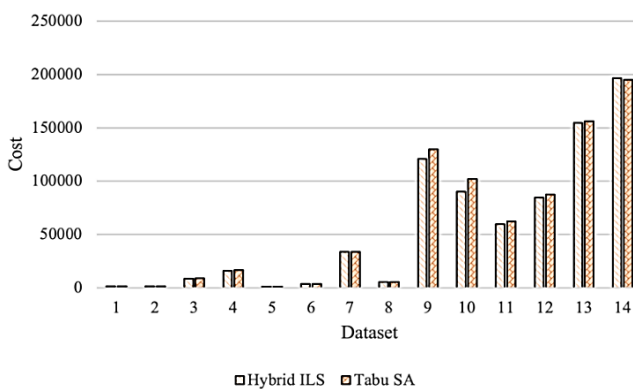


Figure 7. Comparison hybrid ILS and Tabu-SA on average result

**5.4. Result analysis**

Based on section 5.2, exploring solutions impacts the quality of the solutions produced in this case study. The exploring impacts shown from the SA and HC composition of 4:1 produces the best results. SA algorithms that allow accepting worse solutions result in opportunities to generate more exploration. While the HC algorithm only needs a small portion to polish the solution of the SA algorithm.

A large number of explorations runs has a more significant effect on large datasets. Large datasets will have more extensive search space solutions. Hence, applying the HC algorithm too much will cause it to be easily trapped in local optimal conditions. When it is trapped in local optimal conditions, the remaining iterations carried out by the HC algorithm are in vain. Meanwhile, in small and medium data sets, the differences in the composition of SA and HC do not show significant differences.

This study also shows that the hybridization method, by combining several local search algorithms as move acceptance, produces promising results. The result of this hybridization is shown by the superiority of this algorithm in section 5.3. The hybridization scheme developed in this study is different from the research [25]. In research [25], the move acceptance process only uses the SA algorithm. Meanwhile, the TS algorithm is used in other parts to ensure the diversity of the LLH operators being run. In this study, hybridization was carried out by combining three types of local search algorithms to focus on improving solutions in the process of move acceptance.

## 6. CONCLUSION

TSP is an NP-Hard problem which is still an open problem. This research develops an algorithm to solve one of the case studies of the TSP problem, namely TSC 2.0. This problem aims to sort the travel routes modeled in the form of TSP so that the costs are as low as possible. The algorithm developed in this study is a hybridization algorithm between ILS and SA, and HC. Algorithm development is carried out using the hyper-heuristic method. Three types of LLH swaps were implemented in this study. The test results show that this algorithm performs well by optimizing better by 15.7% on average and 11.4% based on the best results from previous studies using the Tabu-SA algorithm.

## APPENDIX

Table 1. Dataset TSC 2.0

Dataset	Number of Area	Number of City	Type
Dataset 1	10	10	Artificial
Dataset 2	10	15	Artificial
Dataset 3	13	38	Artificial
Dataset 4	40	99	Artificial
Dataset 5	46	138	Artificial
Dataset 6	96	192	Artificial
Dataset 7	150	300	Artificial
Dataset 8	200	300	Artificial
Dataset 9	250	250	Artificial
Dataset 10	300	300	Artificial
Dataset 11	150	200	Real
Dataset 12	200	250	Real
Dataset 13	250	275	Real
Dataset 14	3000	300	Real




## 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," *Applied Soft Computing*, vol. 86, Jan. 2020, doi: 10.1016/j.asoc.2019.105887.
- [2] S. Ebadinezhad, "DEACO: Adopting dynamic evaporation strategy to enhance ACO algorithm for the traveling salesman problem," *Engineering Applications of Artificial Intelligence*, vol. 92, Jun. 2020, doi: 10.1016/j.engappai.2020.103649.
- [3] Y. Wang and Z. Han, "Ant colony optimization for traveling salesman problem based on parameters optimization," *Applied Soft Computing*, vol. 107, Aug. 2021, doi: 10.1016/j.asoc.2021.107439.
- [4] C. Jiang, Z. Wan, and Z. Peng, "A new efficient hybrid algorithm for large scale multiple traveling salesman problems," *Expert Systems with Applications*, vol. 139, Jan. 2020, doi: 10.1016/j.eswa.2019.112867.
- [5] O. Cheikhrouhou and I. Khoufi, "A comprehensive survey on the multiple traveling salesman problem: applications, approaches and taxonomy," *Computer Science Review*, vol. 40, May 2021, doi: 10.1016/j.cosrev.2021.100369.
- [6] V. Raman and N. S. Gill, "Review of different heuristic algorithms for solving travelling salesman problem," *International Journal of Advanced Research in Computer Science*, vol. 8, no. 5, pp. 423–425, 2017, doi: 10.26483/ijarcs.v8i5.3319.
- [7] I. G. A. Premananda, A. Tjahyanto, and A. Muklason, "Hybrid whale optimization algorithm for solving timetabling problems of ITC 2019," in *2022 IEEE International Conference on Cybernetics and Computational Intelligence (CyberneticsCom)*, Jun. 2022, pp. 317–322. doi: 10.1109/CyberneticsCom55287.2022.9865647.
- [8] Y. Wang, "The hybrid genetic algorithm with two local optimization strategies for traveling salesman problem," *Computers & Industrial Engineering*, vol. 70, pp. 124–133, Apr. 2014, doi: 10.1016/j.cie.2014.01.015.
- [9] M. Mahi, Ö. K. Baykan, and H. Kodaz, "A new hybrid method based on particle swarm optimization, ant colony optimization and 3-opt algorithms for traveling salesman problem," *Applied Soft Computing*, vol. 30, pp. 484–490, May 2015, doi: 10.1016/j.asoc.2015.01.068.
- [10] P. Stodola, K. Michenka, J. Nohel, and M. Rybanský, "Hybrid algorithm based on ant colony optimization and simulated annealing applied to the dynamic traveling salesman problem," *Entropy*, vol. 22, no. 8, Aug. 2020, doi: 10.3390/e22080884.




- [11] J. A. Soria-Alcaraz, E. Özcan, J. Swan, G. Kendall, and M. Carpio, "Iterated local search using an add and delete hyper-heuristic for university course timetabling," *Applied Soft Computing*, vol. 40, pp. 581–593, Mar. 2016, doi: 10.1016/j.asoc.2015.11.043.
- [12] G. H. G. da Fonseca, H. G. Santos, T. Â. M. Toffolo, S. S. Brito, and M. J. F. Souza, "GOAL solver: a hybrid local search based solver for high school timetabling," *Annals of Operations Research*, vol. 239, no. 1, pp. 77–97, Apr. 2016, doi: 10.1007/s10479-014-1685-4.
- [13] T. Song, S. Liu, X. Tang, X. Peng, and M. Chen, "An iterated local search algorithm for the University Course timetabling problem," *Applied Soft Computing*, vol. 68, pp. 597–608, Jul. 2018, doi: 10.1016/j.asoc.2018.04.034.
- [14] J. Brandão, "A memory-based iterated local search algorithm for the multi-depot open vehicle routing problem," *European Journal of Operational Research*, vol. 284, no. 2, pp. 559–571, Jul. 2020, doi: 10.1016/j.ejor.2020.01.008.
- [15] J. Brandão, "Iterated local search algorithm with ejection chains for the open vehicle routing problem with time windows," *Computers & Industrial Engineering*, vol. 120, pp. 146–159, Jun. 2018, doi: 10.1016/j.cie.2018.04.032.
- [16] G. Macrina, L. Di Puglia Pugliese, F. Guerriero, and G. Laporte, "The green mixed fleet vehicle routing problem with partial battery recharging and time windows," *Computers & Operations Research*, vol. 101, pp. 183–199, 2019, doi: 10.1016/j.cor.2018.07.012.
- [17] C. Archetti, D. Feillet, A. Mor, and M. G. Speranza, "An iterated local search for the traveling salesman problem with release dates and completion time minimization," *Computers & Operations Research*, vol. 98, pp. 24–37, 2018, doi: 10.1016/j.cor.2018.05.001.
- [18] M. M. de Sousa, P. H. González, L. S. Ochi, and S. de L. Martins, "A hybrid iterated local search heuristic for the traveling salesperson problem with hotel selection," *Computers & Operations Research*, vol. 129, May 2021, doi: 10.1016/j.cor.2021.105229.
- [19] Y. Yuan, K. Tole, F. Ni, K. He, Z. Xiong, and J. Liu, "Adaptive simulated annealing with greedy search for the circle bin packing problem," *Computers & Operations Research*, vol. 144, Aug. 2022, doi: 10.1016/j.cor.2022.105826.
- [20] N. Leite, F. Melício, and A. C. Rosa, "A fast simulated annealing algorithm for the examination timetabling problem," *Expert Systems with Applications*, vol. 122, pp. 137–151, May 2019, doi: 10.1016/j.eswa.2018.12.048.
- [21] S. L. Goh, G. Kendall, and N. R. Sabar, "Simulated annealing with improved reheating and learning for the post enrolment course timetabling problem," *Journal of the Operational Research Society*, vol. 70, no. 6, pp. 873–888, Jun. 2019, doi: 10.1080/01605682.2018.1468862.
- [22] R. Bellio, S. Ceschia, L. Di Gaspero, and A. Schaerf, "Two-stage multi-neighborhood simulated annealing for uncapacitated examination timetabling," *Computers & Operations Research*, vol. 132, Aug. 2021, doi: 10.1016/j.cor.2021.105300.
- [23] A.-H. Zhou *et al.*, "Traveling-salesman-problem algorithm based on simulated annealing and gene-expression programming," *Information*, vol. 10, no. 1, Dec. 2018, doi: 10.3390/info10010007.
- [24] Y. Zhang, X. Han, Y. Dong, J. Xie, G. Xie, and X. Xu, "A novel state transition simulated annealing algorithm for the multiple traveling salesmen problem," *The Journal of Supercomputing*, vol. 77, no. 10, pp. 11827–11852, Oct. 2021, doi: 10.1007/s11227-021-03744-1.
- [25] E. D. Ahmad, A. Muklason, and I. Nurkasanah, "Route optimization of airplane travel plans using the tabu-simulated annealing algorithm to solve the Traveling Salesman Challenge 2.0," in *2020 International Conference on Computer Engineering, Network, and Intelligent Multimedia (CENIM)*, Nov. 2020, pp. 217–221. doi: 10.1109/CENIM51130.2020.9297892.
- [26] A. Arram, M. Ayob, and A. Sulaiman, "Hybrid bird mating optimizer with single-based algorithms for combinatorial optimization problems," *IEEE Access*, vol. 9, pp. 115972–115989, 2021, doi: 10.1109/ACCESS.2021.3102154.
- [27] R. Dong, S. Wang, G. Wang, and X. Wang, "Hybrid optimization algorithm based on wolf pack search and local search for solving traveling salesman problem," *Journal of Shanghai Jiaotong University (Science)*, vol. 24, no. 1, pp. 41–47, Feb. 2019, doi: 10.1007/s12204-019-2039-9.
- [28] Q. M. Ha, Y. Deville, Q. D. Pham, and M. H. Hà, "A hybrid genetic algorithm for the traveling salesman problem with drone," *Journal of Heuristics*, vol. 26, no. 2, pp. 219–247, Apr. 2020, doi: 10.1007/s10732-019-09431-y.
- [29] Y. Deng, J. Xiong, and Q. Wang, "A hybrid cellular genetic algorithm for the traveling salesman problem," *Mathematical Problems in Engineering*, vol. 2021, pp. 1–16, Feb. 2021, doi: 10.1155/2021/6697598.

## BIOGRAPHIES OF AUTHORS



**Ahmad Muklason**    Assistant Professor at Data Engineering and Business Intelligence Lab., Department of Information Systems, Faculty of Intelligent Electrical, Electronic Engineering and Informatics, Institut Teknologi Sepuluh Nopember, Surabaya, Indonesia. Received his doctoral degree from School of Computer Sciences, The University of Nottingham, UK, in 2017; Master of Science degree of Computer and Information Sciences Department, Universiti Teknologi Petronas, Malaysia, in 2009; and Bachelor of Computer Science from Institut Teknologi Sepuluh Nopember, Surabaya in 2006. He can be contacted at mukhlason@is.its.ac.id.



**I Gusti Agung Premananda**    received his master degree from Institut Teknologi Sepuluh Nopember, Surabaya, Indonesia in 2021 in information systems; and Bachelor of Computer Science from Institut Teknologi Sepuluh Nopember, Surabaya in 2019. Presently he is taking the doctoral program in Institut Teknologi Sepuluh Nopember, Surabaya, Indonesia in 2021 in information systems. He can be contacted at igustiagungpremananda@gmail.com.