Ant Colony Optimization for K-Independent Average Traveling Salesman Problem

Anonymous author

No Institute Given

Abstract. In this paper, we propose a K-independent average traveling salesman problem (KI-Average-TSP) extended from the TSP. This is an optimization problem that minimizes the weighted sum of the average and standard deviation of K circuits' costs, with mutually independent edges. As a method to solve the KI-Average-TSP, we propose K-independent average ant colony optimization (KI-Average-ACO) extended from the original ACO. KI-Average-ACO moves K ants simultaneously using the following two heuristcs to prevent different circuits from sharing the same edge. The first heuristic uses a degree of possible options representing the number of vertices that an ant can reach from its current vertex. The destination of ants is stochastically determined by this value to reduce the circuit construction failure rate. The second heuristic, named 2-best-opt, uses a greedy algorithm in reconstructing a better path to obtain K circuits if circuit construction fails. Comparison results between the approximate solution obtained using KI-Average-ACO and the solution obtained using a quadratic programming method for binary search show that the number of circuits for KI-Average-ACO was higher, and KI-Average-ACO obtained a better approximate solution than the quadratic programming method.

Keywords: Ant Colony Optimization · Traveling Salesman Problem · Heuristics

1 Introduction

One of the common combinatorial optimization problems is the traveling salesman problem (TSP) [6]. Given distances (edges) connecting cities, the TSP finds the shortest routes to visit all cities. The TSP is applied to various problems, such as vehicle routing and job-shop scheduling [5]. However, optimization problems in the real world are considered more complicated than the TSP. For example, in a transportation company's delivery plans, even if the company constructs the shortest circuits to access cities, these routes may become inaccessible due to road damage or accidents.

In this study, we consider constructing mutually independent circuits with no shared edges to improve reliability and propose a K-independent average TSP (KI-Average-TSP) that minimizes the weighted sum of the average and standard deviations of K circuits' costs, where the circuits are mutually independent, that

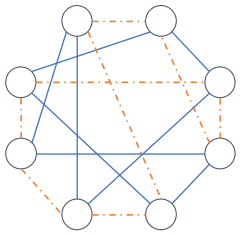

Fig. 1. An example of K-Independent paths (with $K = 2$), where multiple paths do not share the same edge in $N = 8$ complete graph.

is, there are no shared edges among the circuits. Compared to a study to find multiple independent Hamiltonian paths in a graph [11], this study minimizes the cost by considering the standard deviation in a complete graph so that the K circuits act as backup routes. Fig. 1 shows an example of $K = 2$ circuits in a complete graph of $N = 8$ vertices.

To solve the problem, we propose an ant colony optimization (ACO)-based K-independent average ACO (KI-Average-ACO). ACO [2, 7] is known as a metaheuristic solution to the TSP using ant swarm characteristics and simulates the pheromone communication of ants on a graph to find the approximate shortest path. In our optimization algorithm, K ants move simultaneously, making it possible for K ants to use equally favorable edges and reducing the standard deviation. However, as the ants move, the number of reachable vertices for the ants decreases, and the circuit construction failure rate increases. Therefore, we introduce two heuristics to reduce the failure rate of circuit construction. The key concept of the first heuristic is a degree of possible options that represents the number of vertices an ant can reach from its current vertex. The destination of ants is stochastically determined by this number to reduce the circuit construction failure rate. The second heuristic, called 2-best-opt, is an algorithm based on the idea of 2-opt [3, 6]. It uses a greedy algorithm in reconstructing a better path to obtain K circuits if circuit construction fails.

In this paper, we evaluated our proposed method by comparing the approximate solution obtained using KI-Average-ACO with the solution obtained using a quadratic programming method for a binary search. Comparison results showed that the number of circuits for KI-Average-ACO was higher, and KI-Average-ACO obtained a better approximate solution than the quadratic programming method.

Related Work. ACO is a search algorithm that is inspired by the process by which ants use pheromones to discover the shortest path to food [2]. In particular, it is known as metaheuristics for finding approximate solutions to shortest path problems, as typified by the TSP. An application of ACO is the vehicle routing problem [10], which delivers resources from a depot to customers through delivery vehicles. Another application is the multiple TSP [4], which

builds a partial traveling circuit in which multiple salesmen share each other's visits to cities from a depot.

On the other hand, a few studies have been conducted on algorithms to find multiple independent circuits in a complete graph. Teng [11] reported properties such as the conditions for establishing multiple Hamiltonian paths in a given graph. However, algorithms for constructing circuists that are independent of each other have not yet been investigated.

Paper Organization. The organization of this paper is as follows. Section 2 defines KI-Average-TSP and a related problem named KI-Total-TSP. Section 3 introduces our optimization algorithm named KI-Average-ACO. Section 4 presents experimental results. Finally, Section 5 concludes the paper and presents future work.

2 Problem Description

In this section, we give the definition of our target problem named K-indipendent average traveling salesman problem (KI-Average-TSP). In addition, although not directly dealt with in this study, we also define K-independent total traveling salesman problem (KI-Total-TSP), which is another optimization problem related to KI-Average-TSP. The former is the problem of finding the K-independent circuits that minimize the weighted sum of average and standard deviation in a given complete graph, while the latter is the problem that removes the term of standard deviation from the objective function in KI-Average-TSP and minimizes only the total cost of K-independent circuits.

Here, let $G = (V, E, d)$ (with $|V| = N, |E| = \frac{N(N-1)}{2}$) be a weighted undirected complete graph, where d_{ij} is the weight of edge (i, j) and N is the number of cities.

2.1 K-Independent Average TSP

KI-Average-TSP is a problem to minimize the weighted sum of the average and standard deviations of K independent circuits' cost in a graph G. The definition is as follows.

Definition 1. *KI-Average-TSP is a problem to perform the following optimization in graph $G = (V, E)$.*

$$
\begin{aligned}
&\textbf{min} && \mathrm{cost}_{\mathrm{avg}} + \gamma \cdot \mathrm{cost}_{\mathrm{sd}}^{\theta} && && (1)\\
&\textbf{subject to} && \sum_{k \in K} x_{ijk} \leq 1 && (\forall i, j\ (i \neq j)) && (2)\\
& && \sum_{j \in V} x_{ijk} = 1 && (\forall i, k) && (3)\\
& && \sum_{j \in V} x_{jik} = 1 && (\forall i, k) && (4)\\
& && \mathrm{u}_{ik} + 1 - (N-1)(1 - x_{ijk}) \leq \mathrm{u}_{jk} && (\forall i, j, k) && (5)\\
& && x_{ijk} \in \{0, 1\} && && (6)\\
& && 0 \leq u_{ik} \leq N - 1 && && (7)
\end{aligned}
$$

where

$$
\mathrm{cost}_{\mathrm{avg}} = \frac{1}{K} \mathrm{cost}_{\mathrm{sum}}, \tag{8}
$$

$$
\mathrm{cost}_{\mathrm{sd}} = \sqrt{\frac{1}{K}(\sum_{k \in K}(\sum_{i \in V}\sum_{j \in V} d_{ij} \cdot x_{ijk} - \frac{1}{K}(\sum_{i \in V}\sum_{j \in V}\sum_{k' \in K} d_{ij} \cdot x_{ijk'}))^2)}. \tag{9}
$$

The value x_{ijk} represents the probability of ant k using the edge connecting vertex i to vertex j. The value u_{ik} is the arc-constraint to excludes subtours based on the Miller – Tucker – Zemlin formulation[8]. Hereafter, the total cost of the K circuits is expressed as $\mathrm{cost}_{\mathrm{sum}}$. The weighted sum is represented as $\mathrm{cost}_{\mathrm{ssd}} = \mathrm{cost}_{\mathrm{avg}} + \gamma \cdot \mathrm{cost}_{\mathrm{sd}}^{\theta}$, where $\mathrm{cost}_{\mathrm{avg}}$ and $\mathrm{cost}_{\mathrm{sd}}^{\theta}$ respectively represent the average and standard deviation of K circuits' costs. Variables γ, θ are parameters for weighting the average and standard deviation respectively, and the constraints is the same as in KI-Total-TSP.

2.2 K-Independent Total TSP

The KI-Total-TSP is a problem to minimize the total cost in K circuits among the combinations of K independent circuits in the graph G. The definition is as follows.

Definition 2. *KI-Total-TSP is a problem to perform the following optimization in graph $G = (V, E, d)$.*

$$
\textbf{min} \qquad \sum_{i \in V}\sum_{j \in V}\sum_{k \in K} d_{ij} \cdot x_{ijk} \tag{10}
$$

with the same constraint conditions as in the KI-Average-TSP (i.e., Eqs. 2–7).

We would like to note that the problems introduced above are complementary to each other. That is, the solution of KI-Average-TSP is useful when finding K circuits with similar utility values. On the other hand, KI-Total-TSP is useful when finding K circuits with ranked utility values. Specifically, the former is a case, where multiple packets are sent simultaneously through K routes, while the latter is a case, where spare routes should be prepared for a failure of the current route.

3 K-Independent Average ACO

3.1 Overview

In this section, we explain the proposed K-independent average ant colony optimization (KI-Average-ACO) algorithm for solving KI-Average-TSP. Considering that it is difficult to calculate the exact solution to KI-Average-TSP in a feasible time, we propose KI-Average-ACO to obtain an approximate solution.

Unlike the original ACO, KI-Average-ACO averages the cost of K circuits by repeatedly moving K ants along one edge at a time. However, because the construction failure rate increases when K ants move simultaneously, we use two heuristics. The first heuristic uses a degree of possible options, which indicates the feasible vertices the ant can reach. Ants move to vertices with fewer destinations using this index, putting off many potential movable vertices. The second is 2-best-opt, which searches efficiently by reconstructing failed circuits greedily using 2-opt. Using these heuristics, it is possible to reduce the construction failure rate while averaging the moving cost of K circuits, rather than repeating ACO. The pseudocode of KI-Average-ACO is presented in Algorithm 1.

3.2 Simulaneous Movement of Ants

KI-Average-ACO differs from the original ACO in moving K ants along one edge at a time. In the original Ant System [1] and Max-Min Ant System [9], some ants constructs a path. If these algorithms are applied to KI-Average-TSP repeatedly, some ants constructs a circuit using preferable edges greedily. Consequently, the cost of the circuit in the latter half of the construction increases, the shape becomes complicated, and the averaging cannot be satisfied. In contrast, in Line17 in the pseudocode, KI-Average-ACO moves K ants along one edge at a time, so that all ants can use their preferred edges. In Line19, ants are arranged in descending order of moving cost so far after every movement, and the next ants move in this order. As a result, it is possible to perform the averaging of the cost in a greedy manner, because ants, which have consumed bigger moving cost so far, can move to smaller-cost and pheromone-rich edges more preferentially. We would like to note that the cost of an edge used becomes infinite and the pheromone value becomes zero to ensure that it is not used as much as possible again.

Algorithm 1 KI-Average-ACO

```
function ki_average_aco(G, N, K)
    aco(G)                                          ▷ init G's pheromone with Ant System
    ants = [ ]
    for i = 1 to K do
        ants.append(Ant())
    end for
    for t = 1 to T do
        make_tsps(G, N, K, ants)
        2_best_opt(G, K, ants)
        pheromone_update(G, ants)
    end for
end function

function make_tsps(G, N, K, ants)
    for i = 1 to N do
        for j = 1 to K do
            ants[j].move_one_edge(G)                ▷ use the degree of possible options
        end for
        ants.sort(key=ant.cost_sum, reverse=true)   ▷ sort ants in descending order
    end for
end function

function 2_best_opt(G, N, K, ants)
    for i = 1 to K do
        if ants[i].path has duplicated edge then
            ant = ants[i]
            alts = [ ]
            for j = 1 to N do
                for k = 1 to N do
                    at = ant.2_opt(ant.path[j], ant.path[k])
                    if at.path has no contradiction then
                        alts.append(at)
                    end if
                end for
            end for
            alts.sort(key=ant.cost_sum)
            ants[i] = alts[0]                       ▷ use best swap reducing cost
        end if
    end for
end function

function pheromone_update(G, ants)
    if ants has no duplicated edge then
        update(G, ants)
    end if
end function
```

3.3 Heuristic with Degree of Possible Options

KI-Average-ACO moves K ants simultaneously. As a result, close to the K-th movement, there are possibilities that ants may reuse edges used by other ants, thereby increasing the construction failure. Therefore, to increase the number of successes, we introduce the heuristic with the degree of possible options. This is incorporated into the transition probability equation in Line 17 in the pseudocode.

Assume that each of the K ants has already moved t times out of N times, and the set of vertices visited by ant h is U_h. Here, we define $R_h^t(x) \subseteq V$ as a function that returns a set of vertices where ant h can move consistently from vertex x to vertex y. The word "consistent" averages that vertex $y \in R_h^t(x)$ satisfies $y \notin U_h$ and the edge (x, y) is not yet used by other ants $1, 2, \ldots, h-1, h+1, \ldots, K$.

At this time, transition probability equation P_{ij}^h that ant h selects the next vertex j from the vertex i is defined by Eq. 11.

$$P_{ij}^h = \frac{[\tau_{ij}]^\alpha [\eta_{ij}]^\beta}{\sum_{u \in N_h^t(i)} [\tau_{iu}]^\alpha [\eta_{iu}]^\beta} \times \frac{1}{|R_h^t(j)|}, \tag{11}$$

$$\eta_{ij} = \frac{1}{d_{ij}} \tag{12}$$

where τ_{ij} is the pheromone quantity of the edge (i, j) and η_{ij} is the heuristic value. $N_h^t(i)$ is a set of vertices where an ant h can move from vertex i in the t-th move. In Eq. 11, the transition probability is divided by the number of vertices $|R_h^t(j)|$ that ant h can move from vertex j. Using this operation, ants can preferentially move to vertices that have little non-affordably movable vertices. This reduces the construction failure rate. Although the time complexity is increased by $O(N)$ during the moving process, the construction failure rate is reduced and an approximate solution can be more efficiently searched.

3.4 2-best-opt

In Line9, we use the heuristic called 2-best-opt to improve paths after constructing the circuits to further increase the success probability of constructing independent circuits and minimizing cost_{ssd}. Procedures of the 2-best-opt are depicted from Line23 to 40.

Assume that in the K paths after the construction, an edge $e = (a, b)$ is redundantly used in multiple paths of F (> 1) ants. At this time, 2-best-opt is performed for each of the $l_1, l_2, \ldots, l_F$ paths of F ants. 2-best-opt is the following operation and is shown in Fig. 2 and from Line23 to 40.

Consider a case where an edge $e = (a, b)$ is redundantly used by other paths of a certain path l and other $N-1$ edges $e' = (c, d)$ of the path l are exchanged by 2-opt. At this time, if the two new edges $e_1 = (a, c)$ and $e_2 = (b, d)$ are not yet used for any K paths, the number of use of edges can be reduced without contradiction, and the edge e' is added to the replacement candidate set S in

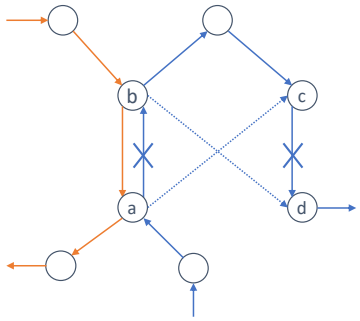

Fig. 2. The example of 2-best-opt which is trying to swap $e = (a, b)$ and $e' = (c, d)$ because the edge e is used twice. Swapped edges $e_1 = (a, c)$, $e_2 = (b, d)$ can reduce usage count in $e = (a, b)$.

Line 30 to 33. Then, among the exchange candidates of these edges, an edge $e' \in S$ is selected and swapped by e in 2-opt, so that the moving cost of the path l becomes the smallest by exchanging with the edge e in Line 36 and 37.

Using this 2-best-opt, the overlapping edges causing the failure can be corrected greedily and the construction failure rate and cost_{ssd} can be reduced.

3.5 Pheromone Update

After performing 2-best-opt, in Line 10, if K paths are reconstructed and are independent of each other, the pheromone update equation, Eq. 13, is executed.

$$\tau_{ij}(t+1) = \rho \cdot \tau_{ij}(t) + \sum_{h=1}^{K} \Delta_{ij}^{h}, \tag{13}$$

$$\Delta_{ij}^{h} = \begin{cases} \frac{1}{C_h + \text{cost}_{\text{sd}}^{\theta}} & ((i,j) \in l_h) \\ 0 & (otherwise) \end{cases} \tag{14}$$

where ρ is the retention rate of the pheromone, C_h is the cost of the circuit constructed by ant h, l_h is the path of ant h, and θ is equal to θ in Eq. 1. However, if the K paths are not independent of each other, the updating equation for the pheromone on the graph and the evaporation of pheromone are not performed from Line 43 to 45. This tries to centralize searching by updating.

4 Empirical Study

4.1 Parameters and Settings

This section presents two experiments to evaluate our algorithm. In the first experiment, the performance of the two heuristics used in KI-Average-ACO was

evaluated. In the second experiment, KI-Average-TSPs were solved using KI-Average-ACO and a combinatorial optimization method to compare their solutions and the time taken to obtain these solutions.

Table 1. Parameter Settings.

Parameter	Value
α (pheromone rate)	1
β (heuristic rate)	3
ρ (pheromone residual rate)	0.97
Number of ants	N
AT (number of the cycles in Ant System)	200
KT (number of the cycles in KI-Average-ACO)	1000
LT (number of trials)	10

The parameters in the experiments are as shown in Table 1. Here, AT represents the number of cycles to initialize the pheromone using the Ant System on the graph before starting KI-Average-ACO. KT represents the number of cycles that KI-Average-ACO executes to build K paths. LT represents the number of experimental trials performed.

4.2 Performance Evaluation of Heuristics

To evaluate the performance of the heuristics used in KI-Average-ACO, we considered four cases with and without each heuristic and compared them by solving KI-Average TSP for each case. In every case, the pheromone was always updated.

As the graphs used for the optimization problems, we selected three graphs ulysses22, bays29, and att48, which have a different number of vertices from TSPLIB, the dataset often used for performance evaluations of TSP solution algorithms.

We attempted to construct $K = 6$ circuits in each graph. We measured the weighted cost $\mathrm{cost}_{\mathrm{ssd}}$, execution time, and construction failure rate. Even if only one edge is duplicated, this is considered as a construction failure. For example, if the construction failure rate is 0.48, this means that one or more edges have been used in duplicates in 480 of 1,000 trials. The results of the above experiments are shown in Table 2. Here, DPO, 2BO, DPO+2BO, and NONE represent cases, where only the heuristic with degree of possible options was used, only 2-best-opt was used, both two heuristics were used, and neither of the two was used, respectively.

Table 2 shows the comparison results of the heuristic performance in KI-Average-ACO. These results show that 2-best-opt, in particular, improves the circuit construction failure rate in all three graphs. Moreover, the execution time of 2-best-opt is not long, so this heuristic would be feasible. On the other hand, heuristic with the degree of possible options did not contribute to reduce the cost

Table 2. Performance of Heuristics in KI-Average-ACO.

Problem	Heuristics	Cost of solution	Required time	Failure rate
ulysses22	NONE	6.39×10^4	14	0.999
	DPO	$\mathbf{5.22 \times 10^4}$	55	0.999
	2BO	5.33×10^4	**16**	**0.0**
	DPO+2BO	5.84×10^4	57	**0.0**
bays29	NONE	1.15×10^4	32	0.994
	DPO	1.01×10^4	126	0.970
	2BO	$\mathbf{8.97 \times 10^3}$	**34**	**0.0**
	DPO+2BO	9.83×10^3	126	**0.0**
att48	NONE	2.96×10^5	70	0.989
	DPO	4.12×10^5	488	0.964
	2BO	$\mathbf{2.91 \times 10^5}$	**72**	**0.0**
	DPO+2BO	3.43×10^5	489	**0.0**

of the solution even though the execution time was longer than that of 2-best-opt. This is because the execution time of this heuristic takes $O(N)$, and costly edges were also selected to give priority to the reduction of the circuit construction failure rate. However, for ulysses22, the cost of solution was the smallest when heuristic with the degree of possible options was used. This suggests that this heuristic may also be useful depending on the problem type.

4.3 Performance Evaluation of KI-Average-ACO

We evaluated the performance of KI-Average-ACO. Specifically, we compared the costs of solutions obtained using KI-Average-ACO to that obtained using a combinatorial optimization algorithm. In the experiments, we used the graph gr17 with $N = 17$ as a problem to solve. The weighted parameters γ and θ of the KI-Average-TSPs were set as 1 and 2, respectively.

In general, it is difficult to find the exact solution for KI-Average-TSP. Indeed, in our preliminary experiments, we observed that exact solutions could be found only for problems, where the size of the graph is $N < 8$ and the number of circuits is $K < 3$ in a feasible time. Therefore, we evaluated the performance of KI-Average-ACO and the combinatorial optimization algorithm by comparing the costs of approximate solutions instead of exact solutions. The combinatorial optimization algorithm used as the comparison target is as follows. Since the standard deviation can be calculated from K circuits, binary search was used to determine whether the maximum difference d of costs between the K circuits could be less than or equal to a certain threshold value. Repeating this operation, while gradually lowering the threshold value within a predetermined time limit, minimized the maximum difference in the costs of K circuits (thus, the standard deviation was also minimized). In this case, the time limit was set to 300 seconds.

Figure 3 shows the experimental results. The horizontal and vertical axes of the graph represent the K number of circuits in the problem and the total costs

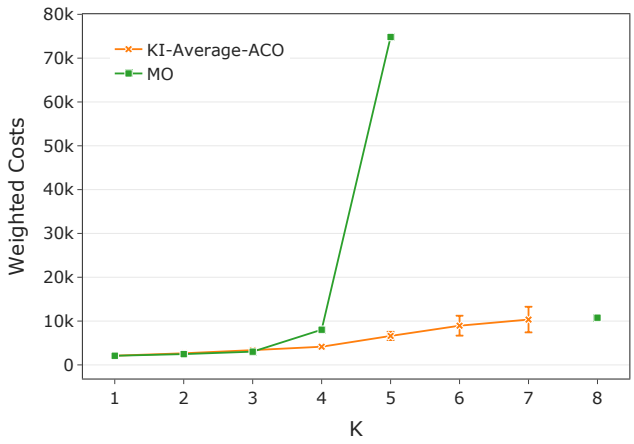

Fig. 3. Costs of Solutions for KI-Average-TSP.

of the solutions, respectively. "KI-Average-ACO" and "MO" represent the results obtained using our algorithm and the mathematical optimization algorithm, respectively. Here, the plots of MO for $K = 6, 7$ are lost because MO did not find any solution within the time limit. This figure shows that for $K = 1, 2, 3$, the total cost of the solutions obtained by the proposed algorithm was almost the same as that of MO. Furthermore, for $K = 4, 5, 6, 7$, our algorithm obtained better solutions. We would like to remark on the result when $K = 8$. It is considered that the reason why MO obtained a solution in this case is that the solution has to use all edges, so it is no longer necessary to decide whether or not to use edges in the solution. On the other hand, this did not bring any benefit to the solution of our algorithm even though it took more time to calculate as the problem became more complicated. As a result, no solution could be found. However, this is a special case, and overall, we have observed that it is useful, especially when the problems are complicated compared to the mathematical optimization algorithm.

5 Conclusions and Future Work

In this study, we proposed two problems named KI-Average-TSP and KI-Total-TSP, which were extensions of the TSP. We also proposed KI-Average-ACO, an optimization algorithm to solve KI-Average-TSP. The idea behind our algorithm is to move K ants simultaneously. However, to reduce the failure rate of circuit construction, we introduced two heuristics with the degree of possible options and 2-best-opt, a heuristic based on 2-opt. We evaluated the performance of our algorithm and the effectiveness of the heuristics. In the experiments, we observed that 2-best-opt significantly contributed to reducing solution costs and construction failure rates. In addition, the solution cost of KI-Average-ACO was

reduced when the circuit to be constructed was larger compared to the solution cost of the mathematical optimization algorithm for binary search.

One of the important future directions is to apply our algorithm to realistic problems. In particular, we would like to generalize KI-Average-TSP to formulate an optimization problem that allows $m < K$ edges to be shared between circuits and to find its solution algorithm.

References

1. Dorigo, M., Maniezzo, V., Colorni, A.: Ant system: optimization by a colony of cooperating agents. IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) **26**(1), 29–41 (1996)
2. Dorigo, M., Stützle, T.: Ant colony optimization: overview and recent advances. In: Handbook of metaheuristics, pp. 311–351. Springer (2019)
3. Johnson, D.S., McGeoch, L.A.: The traveling salesman problem: A case study in local optimization. Local search in combinatorial optimization **1**(1), 215–310 (1997)
4. Junjie, P., Wang, D.: An ant colony optimization algorithm for multiple travelling salesman problem. pp. 210–213 (01 2006). https://doi.org/10.1109/ICICIC.2006.40
5. Lenstra, J.K., Kan, A.R.: Some simple applications of the travelling salesman problem. Journal of the Operational Research Society **26**(4), 717–733 (1975)
6. Lin, S., Kernighan, B.W.: An effective heuristic algorithm for the traveling-salesman problem. Operations research **21**(2), 498–516 (1973)
7. Matai, R., Singh, S.P., Mittal, M.L.: Traveling salesman problem: an overview of applications, formulations, and solution approaches. Traveling salesman problem, theory and applications **1** (2010)
8. Pataki, G.: Teaching integer programming formulations using the traveling salesman problem. SIAM review **45**(1), 116–123 (2003)
9. Stützle, T., Hoos, H.H.: Max–min ant system. Future generation computer systems **16**(8), 889–914 (2000)
10. Tan, W.F., Lee, L.S., Majid, Z.A., Seow, H.V.: Ant colony optimization for capacitated vehicle routing problem. Journal of Computer Science **8**(6), 846–852 (2012)
11. Teng, Y.H., Tan, J.J., Ho, T.Y., Hsu, L.H.: On mutually independent hamiltonian paths. Applied Mathematics Letters **19**(4), 345 – 350 (2006). https://doi.org/https://doi.org/10.1016/j.aml.2005.05.012, `http://www.sciencedirect.com/science/article/pii/S0893965905002387`