IIAI Open Conference Publication Series
IIAI Letters on Informatics and Interdisciplinary Research
Vol.004, LIIR175
DOI: https://doi.org/10.52731/liir.v004.175

Improvement of a Greedy-Based TSP Heuristic and Its Application to ACO

Takuya Sugimoto [*], Noboru Abe [*], Kazuaki Yamaguchi [†]

Abstract

Recent years have observed increased demand for deliveries and a shortage of human resources, necessitating more efficient transportation paths for products and other items in transportation, logistics, and associated sectors. This study introduces a heuristic by improving the greedy-based algorithm to solve the traveling salesman problem. This proposed method can rapidly find numerous promising paths. Furthermore, we explored the application of the proposed method for pheromone deposition in ant colony optimization (ACO). Specifically, pheromones are deposited on the paths charted by ants and those investigated using the proposed method. Computational experiments provided robust evidence of the effectiveness of the proposed method and its integration with ACO.

Keywords: ant colony optimization, greedy-based heuristic, traveling salesman problem.

1 Introduction

The traveling salesman problem (TSP), a graph-theoretic problem, is widely used in various applications, especially in transportation and logistics. Based on the recent surge in delivery demand, triggered by factors such as online shopping, shortage of delivery personnel has emerged as a serious problem. For a company to deliver packages with minimal people, efficient transportation of packages to the delivery location is necessary. Hence, identification of an effective solution for the TSP becomes imperative. However, as TSP is an NP-hard problem [1], identification of an optimal solution necessitates substantial computation time. Thus, numerous heuristic algorithms have been proposed for TSP to date.

This study introduces a heuristic algorithm by improving the existing greedy-based algorithm for TSP. Additionally, we explore the integration of the proposed algorithm with pheromone deposition in ant colony optimization (ACO), a nature-inspired stochastic metaheuristic that often applied to TSP [2][3][4][5][6][7].

2 Preparation

2.1 Traveling salesman problem

The traveling salesman problem (TSP) is a problem for finding an optimal cycle with minimum weight across a given graph such that each vertex is visited exactly once. Figure 1 shows an example. Here, the optimal path is represented by thick lines. Since TSP is an NP-

[*] Osaka Electro-Communication University, Osaka, Japan
[†] Kobe University, Hyogo, Japan

hard problem, several heuristic algorithms have been proposed to solve it in the literature, and it continues to be an area of active study. Furthermore, problems derived from TSP, such as dynamic TSP (DTSP) [3][5][8], multiple TSP [7], and multiobjective TSP [9], are under active investigation.

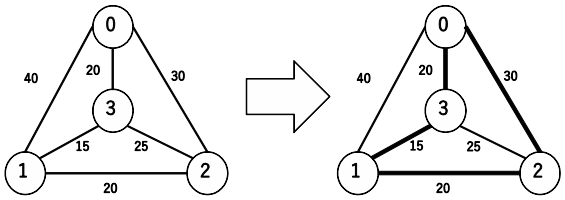

Figure 1: Example of TSP and its optimal path

2.2 Distributed greedy TSP (Dg-TSP) method

Sharma and Chou [10] recently proposed four distinct methodologies for the DTSP. One such method, Dg-TSP, is based on the greedy algorithm, and it can rapidly explore various spectrum of paths. Initially, Dg-TSP investigates all edges connected to the starting vertex. For subsequent vertices, Dg-TSP considers only a single outgoing edge with the least weight connected to each vertex. Thus, for n vertices, Dg-TSP explores n-1 paths. Figure 2 shows an example of Dg-TSP exploration for four vertices 0, 1, 2, and 3, where the edges explored by Dg-TSP are depicted as thick lines.

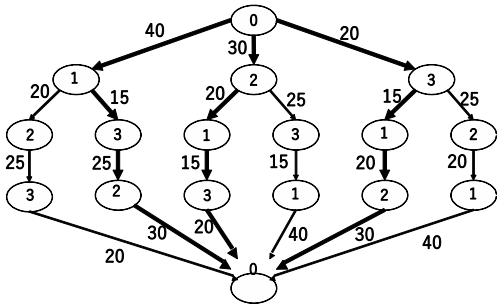

Figure 2: Example of explored edges by Dg-TSP

2.3 Ant colony optimization (ACO)

ACO belongs to swarm-based metaheuristics, a stochastic solution method for computational problems. ACO employs a multitude of artificial ants—acting as information processing units—to construct solutions for optimization problems using graph-based paths. The transfer of pheromone information with each iteration enables ACO to lean toward efficient paths, thereby generating superior solutions. The underlying algorithm is as follows:

1. Initialize the pheromone value of each edge to $init$.
2. Repeat steps i.–iv. until the number of iterations reaches a predefined number.

i. Have each ant explore a path stochastically based on the pheromone amount on the edges and edge weights.

ii. For each path found by each ant, determine the pheromone deposition amount.

iii. Evaporate pheromone according to evaporation factor ρ.

iv. Deposit pheromone on each path determined in step ii.

3. Output the best solution found.

The pheromone amount on a path is calculated by $Q/length$, where Q represents the reference amount of pheromone and $length$ corresponds to the path length. Notably, a smaller length generates a higher pheromone amount. Furthermore, the selection probability P_j of an outgoing edge (k, j) from vertex k is derived from equation (1) and is selected in alignment with the roulette wheel principle:

$$P_j = \frac{W_{kj}^{-\alpha} \cdot F_{kj}^{\beta}}{\sum_{i \in U} W_{ki}^{-\alpha} \cdot F_{ki}^{\beta}} \text{ for all } j \in U, \quad (1)$$

where U denotes the set of unvisited vertices; α and β represent the parameters controlling the influence of the edge weights and pheromone values, respectively; W_{kj} indicate the weight of edge (k, j); and F_{kj} indicates the pheromone trail deposited on edge (k, j). Note that the edge (k, j) with smaller W_{kj} and/or larger F_{kj} are more likely to be selected.

To date, numerous enhanced methodologies have been proposed for ACOs [2][3][4].

3 Proposed Method and Its Application to ACO

Sharma and Chou [10] proposed Dg-TSP can explore various paths quickly. This paper introduces IDg-TSP by improving Dg-TSP to increase the number of explored paths. Unlike Dg-TSP, which only explores the outgoing edges with the smallest weight for the second vertex, IDg-TSP also considers outgoing edges with relatively low weight. More specifically, for each second vertex k, IDg-TSP explores up to $n/10$ outgoing edges (k, j) with weight less than $wmin_{(k,\,j)}$ + ($wmax_{all}$ - $wmin_{all}$)/10 in ascending order of weight. Here, n refers to the number of vertices, $wmax_{all}$ and $wmin_{all}$ are respectively the maximum and minimum edge weights among all edges, and $wmin_{(k,\,j)}$ is the minimum edge weight among outgoing edges from vertex k to unvisited vertex j. Conversely, for the third vertex and onwards, IDg-TSP, similar to Dg-TSP, only explores the minimum weight outgoing edge to avoid an excessive increase in execution time.

Figure 3 provides examples of the selection of the outgoing edges by the IDg-TSP from the second vertex for $wmax_{all}$ = 25 and $wmin_{all}$ = 4, and the number of vertices is six (ranging from 0 to 5). The edges selected for exploration are shown using thick lines. Figure 3(a) demonstrates an example where the number of edges to be explored increases, whereas Figure 3(b) presents a scenario where it remains unchanged.

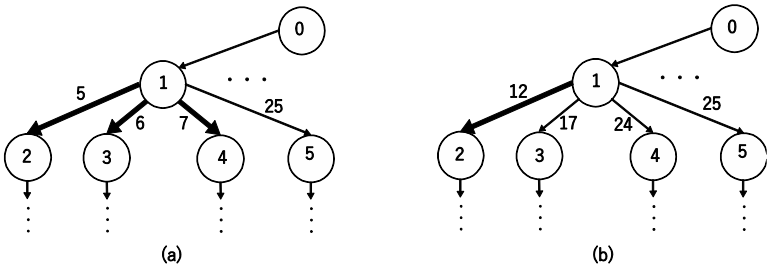

Figure 3: Examples of IDg-TSP's selection of the outgoing edges

Here we propose ACO-IDg, a method wherein IDg-TSP is employed for pheromone deposition in ACO. Using IDg-TSP, pheromones are deposited not only on the paths discovered by the ants but also on those explored by IDg-TSP at every specific number of iterations m. Moreover, pheromones are deposited on the paths explored by IDg-TSP during the initialization step. The amount of pheromone to be deposited in each path by IDg-TSP is determined by $Q/length$, mirroring the paths discovered by the ants. This leads to a modification of steps 1 and iv of ACO as outlined in Section 2.3, resulting in steps 1′ and iv′ below, respectively:

1′. Initialize the pheromone value of each edge to $init$. Additionally, deposit pheromone on the paths explored by IDg-TSP.

iv′. Deposit pheromone on each path determined in step ii. If the number of iterations is a multiple of m, deposit pheromone on the paths explored by IDg-TSP too.

These modifications were made with the objective of enticing ants toward the promising paths explored by IDg-TSP.

4 Experiments

The efficiency of IDg-TSP and ACO-IDg was validated using computational experiments, which were conducted using the C programming language on an AMD Ryzen 7 5700G processor. Parameters ρ, α, and β presented in Section 2.3 were set to 0.9, 1, and 2, respectively. The number of ants was equal to the number of vertices, with the iteration count being 3,000. These parameter values align with those used in [2][6]. As a preliminary

experiment, we examined several values for each parameter ρ, α, and β, and the values mentioned above were generally the best for ACO. The reference amount of pheromone Q was set to 100 to prevent either the edge weight or amount of pheromone in equation (1) from becoming exceedingly dominant. The initial pheromone value $init$ was set to 0.1, and m used in ACO-IDg was set to 50.

Three types of graphs were utilized as input. The first comprised 20 different graphs with 100 vertices, where edge weights vary randomly between 5 and 25. The second and third were benchmark problems berlin52 and kroA100 from TSP-LIB [11], respectively. We prepared the following three conditions for input of graphs with random edge weights:

Condition 1 Each edge weight is given a value between 5 and 25 with a uniform probability.

Condition 2 The probability of a value between 5 and 15 given as an edge weight is twice that of a value between 16 and 25. In other words, the proportion of edges with low weights is high.

Condition 3 The probability of a value between 16 and 25 given as an edge weight is twice that of a value between 5 and 15. In other words, the proportion of edges with high weights is high.

Table 1 shows the results of Dg-TSP, IDg-TSP, ACO, and ACO-IDg for graphs with randomly-determined edge weights based on Condition 1. Each percentage in the table indicates the average ratio of the path length by each method to that by Dg-TSP. Furthermore, Table 2 shows the average results of 10 trials of ACO and ACO-IDg for the two benchmark problems. The average execution times of Dg-TSP, IDg-TSP, ACO, and ACO-IDg of 20 graphs with 100 vertices and random edge weights based on Condition 1 were 0.011, 0.089, 96.99, and 94.19 seconds, respectively. ACO's and ACO-IDg's average execution times for berlin52 of 10 trials were 13.76 and 13.53 seconds, respectively, and those for kroA100 of 10 trials were 96.01 and 94.91, respectively. The proposed IDg-TSP method was slower than Dg-TSP, but it was notably faster, and it can find superior paths than Dg-TSP. Another proposed method, ACO-IDg was nearly as fast as ACO, identifying better paths than ACO for many graphs. For the 20 graphs with 100 vertices and random edge weights based on Condition 1, ACO-IDg outperformed ACO for 12 graphs, but it was inferior for 5 graphs. Under Conditions 2 and 3, ACO-IDg also generally outperformed ACO. The difference between the two methods was larger under Condition 2 than Condition 1 and smaller under Condition 3 than Condition 1. This suggests that ACO-IDg can efficiently utilize edges with low weights.

Table 1: Results for graphs with 100 vertices and random edge weights.

Dg-TSP	IDg-TSP	ACO	ACO-IDg
100.0%	98.3%	96.1%	95.9%

Table 2: Results for benchmark problems berlin52 and kroA100.

		ACO		ACO-IDg	
Graph	Optimum	Best length	Ave. length	Best length	Ave. length
berlin52	7542	7813.74	7849.51	7641.79	7763.80
		103.6%	104.1%	101.3%	102.9%
kroA100	21282	23069.35	23388.14	22587.84	22830.10
		108.4%	109.9%	106.1%	107.3%

5 Conclusions

This study introduced a TSP heuristic algorithm by improving the greedy-based algorithm proposed by Sharma and Chou. This method succeeded in rapidly identifying various promising paths. Additionally, the proposed TSP algorithm was employed for pheromone deposition in ACO. The effectiveness of these methods was validated using computational experiments. Because ACO is frequently utilized in conjunction with local search, future work will aim to investigate the behavior of ACO-IDg in such cases. Furthermore, the application of the proposed methods to DTSP is also a future work. For the latter one, we would like to consider real-time processing on low-performance computers.

References

[1] D. J. Rosenkrantz, R. E. Stearns, P. M. Lewis, “Approximate algorithms for the traveling salesperson problem,” In Proc. the 15th Annual Symp. Switching and Automata Theory (SWAT ‘74), New Orleans, USA, 1974, pp. 33.

[2] R. Skinderowicz, “Improving Ant Colony Optimization Efficiency for Solving Large TSP Instances,” Applied Soft Computing, vol.120, 2022, pp. 108653.

[3] 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, 2020, pp. 884.

[4] H. Ismkhan, "Effective Heuristics for Ant Colony Optimization to Handle Large-scale Problems," Swarm and Evolutionary Computation, vol. 32, 2017, pp. 140.

[5] M. Mavrovouniotis, S. Yang, M. Van, C. Li, and M. Polycarpou, "Ant Colony Optimization Algorithms for Dynamic Optimization: A Case Study of the Dynamic Travelling Salesperson Problem," IEEE Computational Intelligence Magazine, vol. 15, no.1, 2020, pp. 52.

[6] T. Stützle and H.H. Hoos, "MAX-MIN Ant System," Future Generation Computer Systems, vol. 16, no. 8, 2000, pp. 889.

[7] P. Junjie and W. Dingwei, "An Ant Colony Optimization Algorithm for Multiple Traveling Salesman Problem," In Proc. the First Int'l Conf. on Innovative Computing, Information and Control–Volume I (ICICIC'06), Beijing, China, 2006, vol. 1, pp. 210.

[8] L. Strąk, R. Skinderowicz, U. Boryczka, and A. Nowakowski, "A Self-Adaptive Discrete PSO Algorithm with Heterogeneous Parameter Values for Dynamic TSP," Entropy, vol. 21, no. 8, 2019, pp. 738.

[9] P.M. de las Casas, R. Borndörfer, L. Kraus, and A. Sedeño-Noda, "An FPTAS for Dynamic Multiobjective Shortest Path Problems," Algorithms, vol. 14, no. 2, 2021, pp. 43.

[10] S. Sharma and J. Chou, "Distributed and Incremental Travelling Salesman Algorithm on Time-Evolving Graphs," The Journal of Supercomputing, vol. 77, 2021, pp. 10896.

[11] TSPLIB, http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/ (accessed on 22 July 2023).