Journal of Information System and Technology Research

journal homepage: https://journal.aira.or.id/index.php/jistr/

Efficient Route Planning in Northeast Thailand Using a Random Shuffle Enhanced 2-opt Algorithm

Pornsiri Khumla[1], Kamthorn Sarawan[2,*]

[1]Department of Logistics Engineering and Transportation Technology, Kalasin University , Thailand..
[2]Department of Computer Engineering, Kalasin University, Thailand.

ARTICLE INFO

Article history:

Received Dec 26, 2024
Accepted Jan 26, 2025
Available online Jan 31, 2025

Keywords:

2-optRS
Improve 2-opt
Route Planning
Northeast region of Thailand
Traveling Salesman Problem

ABSTRACT

This research presents a detailed comparative analysis of algorithms designed to solve the Traveling Salesman Problem (TSP) within the Northeast region of Thailand, encompassing 20 provinces. The study evaluates the performance of several heuristic algorithms, including Nearest Neighbor, Farthest Insertion, traditional 2-opt, and a novel variant termed 2-optRS, which integrates a random shuffle mechanism. The primary aim is to minimize the total travel distance while considering the computational trade-offs of these algorithms. Authentic distance data between provinces were sourced from Google Maps to ensure accuracy and relevance to real-world applications. The study addresses the dual challenge of computational efficiency and solution quality, emphasizing the practical implications of algorithmic enhancements in route optimization. Among the tested algorithms, 2-optRS exhibited superior performance, achieving a notable reduction of approximately 158.6 km in travel distance compared to the traditional 2-opt algorithm. However, this improvement came at the expense of significantly increased processing time, highlighting the inherent trade-offs between computational demands and optimization benefits. By focusing on the Northeast region, which is poised for significant development through frameworks like the Greater Mekong Subregion (GMS) and Lancang-Mekong Cooperation (LMC), this research provides valuable insights for optimizing logistical networks in this rapidly growing area. The findings underscore the potential of advanced heuristic methods to enhance decision-making in regional transportation planning, contributing to the broader advancement of TSP optimization methodologies.

Corresponding Author:

Kamthorn Sarawan
Department of Computer Engineering, Faculty of Engineering and Industrial Technolog,
Kalasin University, Thailand
Email: kamthorn.sa@ksu.ac.th

1. INTRODUCTION

The Traveling Salesman Problem (TSP) [1-4] is a well-known combinatorial optimization issue that has received a great deal of interest from the fields of operations research and computer science. The TSP's goal is to identify the quickest path that enables a traveling salesman to visit each city exactly once before returning to the starting city. The TSP poses considerable computational challenges due to its classification as an NP-hard problem [3, 5-6]. The number of possible solutions grows factorially with the number of cities, making it computationally infeasible to find the optimal solution for large-scale instances. Therefore, researchers have developed various algorithms and approaches to approximate the optimal solution or find near optimal solutions within a reasonable computational time.

Previous research on TSP has primarily focused on specific problem domains, or often relying on standardized datasets for experimentation and evaluation. Zeravan Arif Ali et al [7] presents a hybrid genetic algorithm that combines genetic and local search algorithms to solve the TSP and achieve optimal or near-optimal solutions. The algorithm improves the crossover and mutation operators of the genetic algorithm and uses local search to find the best local solutions. Numerical

outcomes show that the proposed algorithm is more effective than the conventional genetic algorithm in attaining optimal or near-optimal solutions. The algorithm is tested on various TSP problems from the TSP library and demonstrates its ability to achieve outcomes close to the optimal solution. Otman and Jaafar [8] present a comparative study of adaptive crossover operators for genetic algorithms in TSP. They experiment with different operators and conclude that the OX operator is the most suitable for resolving the TSP problem. Zhou et al. [9] addresses the Multiple Traveling Salesman Problem (MTSP) and proposes two Partheno Genetic Algorithms (PGA) to solve it. They introduce new mutation operations and evaluate the algorithms using TSPLIB benchmarks, demonstrating the superiority of the IPGA algorithm. The fourth paper by Ishaya et al. investigates two methods for solving TSP Branch and Cut (Exact Method) and the 2-opt machine learning method (Heuristics algorithm). They find that the branch-and-cut algorithm provides an optimal solution, while the machine learning method gives a near-optimal solution with a small optimal gap.

This study focuses on addressing the challenges of optimizing travel routes in the Northeast region of Thailand, a geographically diverse area comprising 20 provinces. Despite its strategic importance in regional and national development through frameworks like the Greater Mekong Subregion (GMS), Aeyawadee-Chaopraya-Mekong Economic Cooperation (ACMECS), and Lancang-Mekong Cooperation (LMC) [10], limited research has been conducted on applying advanced heuristic algorithms to optimize travel routes in this specific context. Previous studies on the Traveling Salesman Problem (TSP) have largely utilized standardized datasets or focused on general algorithmic improvements without contextualizing their applications in real-world, region-specific cases. Furthermore, existing research rarely addresses the balance between computational efficiency and solution quality when solving TSP for practical transportation networks.

To bridge this gap, this study evaluates multiple heuristic algorithms including the Nearest Neighbor, Farthest Insertion, traditional 2-opt, and an enhanced 2-optRS variant with a random shuffle mechanism using authentic distance data from Google Maps. By applying these algorithms to real-world data, this research not only provides a comparative analysis of their performance but also highlights the trade-offs between solution optimization and computational demands. The findings aim to contribute valuable insights for practical decision-making [11] in logistical planning while advancing algorithmic methodologies tailored to specific geographical and developmental contexts.

2. RESEARCH METHOD

The research methodology employed in this study involves a systematic approach consisting of three key steps: data collection and pre-processing, algorithm analysis and performance evaluation. Figure 1 illustrates the detailed procedure followed throughout the research process.

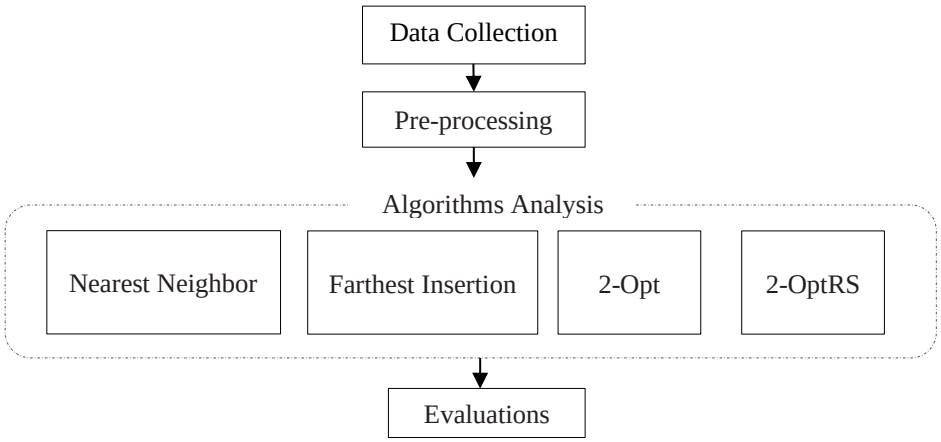


Figure 1. A flow chart of the methodology

2.1. Data Collection and Preprocessing

This study has determined the size of the sample, which includes 20 provinces in Thailand's northeast. Google Maps [12] was used to gather data to gather pertinent information. The acquired dataset consists of the distances between neighboring provinces, indicating the existence of routes and distances that are used to connect adjacent provinces only within each province. For example, NONG KHAI -> (BUENG KAN, UDON THANI, LOEI, SAKON NAKHON). Presenting a geographical representation of the provinces located in the northeastern region, accompanied by a graphical illustration of the spatial relationships and distances between these provinces and their adjacent territories, as delineated in Figure 2 [13].

Then in the process of obtaining distance information between provinces, the collected data was transformed into matrix form. This transformation allowed us to organize the distance data systematically and comprehensively. By representing the distances between provinces in matrix form, we were able to discern the possible distances of all connected provinces. In this step, Dijkstra's algorithm [14-15] was employed. This algorithm is specifically designed to determine the shortest distance between a given starting point and any other point within a graph. By iteratively exploring the shortest paths through each

vertex, the algorithm effectively identifies the path with the minimum overall distance until reaching the desired destination. An example of a route from CHAIYAPHUM -> KALASIN, which is a non-contiguous province. The shortest path calculated by Dijkstra's algorithm is equal to 202.8 Km. Which is derived from the path: CHAIYAPHUM -> KHON KAEN; 123 Km and KHON KAEN -> KALASIN 79.8 Km.

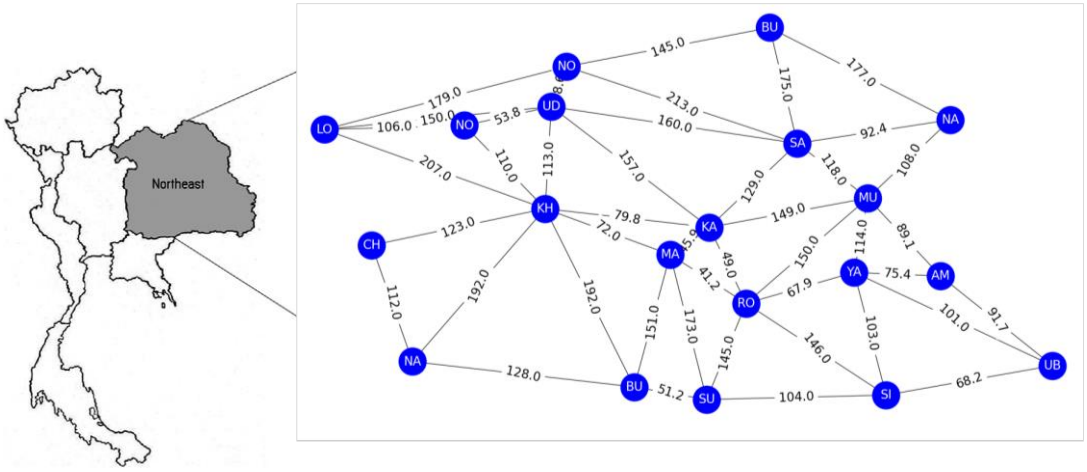


Figure 2. The spatial interconnectivity and relative distances between adjacent provinces.

2.2. Algorithms

The TSP can be addressed using a variety [5, 16-18] of exact algorithms, heuristic algorithms, metaheuristic algorithms, approximation algorithms, and reinforcement learning algorithms, among others. Within the scope of this study, our focus lies on evaluating the performance and effectiveness of different heuristic algorithms as potential solutions for the TSP. By comparing and analyzing the outcomes produced by these heuristic algorithms [18], we aim to gain insights into their respective capabilities and suitability for solving the TSP in the context of our research.

2.2.1 Nearest Neighbor

The Nearest Neighbor (NN) [19-21] algorithm is a straightforward and intuitive heuristic approach widely utilized for solving the TSP. Characterized by its simplicity, the NN algorithm begins its process by selecting an arbitrary starting city. From there, it continuously identifies the nearest unvisited city to the current location, designating this city as the subsequent destination. This procedure is repeated iteratively until all cities in the dataset have been visited once, thus completing the tour. Due to its greedy nature, this method swiftly constructs a tour but does not inherently ensure that the resulting path is the shortest possible, often yielding solutions that are locally but not globally optimal.

Despite its limitation in achieving global optimality, the Nearest Neighbor algorithm is highly valued for its utility in providing rapid solutions, particularly beneficial in scenarios where time constraints are a significant factor. Additionally, its uncomplicated implementation makes it an ideal baseline algorithm for comparative studies with other more complex TSP-solving techniques. The algorithm's performance tends to be more effective when applied to smaller or moderately sized problems, where the proximity of choices doesn't drastically deviate from the global optimum. The general pseudo-code for implementing the Nearest Neighbor algorithm in this study is structured as follows.

Algorithm 1: Nearest Neighbor Algorithm

```
1   pseudo tsp_nearest_neighbor(distance matrix):
2       n = get the number of cities from the distance matrix
3       unvisited(cities) = set of cities from 1 to n (excluding 0)
4       current(city) = 0
5       path = [current]
6       total = 0
7       while unvisited cities are not empty:
8           next(city) = city in unvisited minimum distance from current
9           remove next city from unvisited
10          append next city to path
11          add the distance between current and next to total distance
12          update current city to next city
13      add distance between the last and the starting (0) to total
14      append the starting city (0) to path
15      return path, total distance
16  end pseudo
```

2.2.2 Farthest Insertion

The Farthest Insertion [16, 22] algorithm is a sophisticated heuristic technique employed to address the TSP. This algorithm is distinguished by its strategy of incrementally constructing a tour by specifically selecting the geographically farthest city from the current route. The process begins with an initial simple solution that includes only a single city. In subsequent iterations, the algorithm evaluates all cities not yet included in the tour and identifies the city that is farthest from the current tour's path. This selected city is then inserted into the existing tour at a position where it causes the least increase in the overall tour length, thereby ensuring efficiency in route expansion.

This iterative insertion continues until every city has been incorporated into the tour, thus completing the circuit. The Farthest Insertion algorithm is particularly noted for its ability to produce highly efficient and often near-optimal solutions for the TSP. By always selecting the farthest city, the algorithm effectively minimizes the risk of creating suboptimal local loops that are common in simpler heuristics like the Nearest Neighbor. This method ensures a broader exploration of the solution space, which often results in more effective overall tour configurations compared to those constructed using other heuristic approaches.

The pseudo-code for the Farthest Insertion algorithm, as utilized in this study, can be outlined in the following steps.

Algorithm 2: Farthest Insertion Algorithm

```
pseudo tsp_farthest_insertion(distance matrix):
    n = get the number of cities from the distance matrix
    unvisited(cities) = set of cities from 1 to n (excluding 0)
    current(city) = 0
    path = [current]
    total = 0
    while unvisited is not empty:
      farthest(city) = city in unvisited maximum distance from current
      closest(city) = city in path with the minimum to farthest
       index = index of closest city in path
       insert farthest city after index in path
       current city = farthest city
       remove farthest city from unvisited
    add distance between the last and the starting (0) to total
    append the starting city (0) to path
    return path, total distance
end pseudo
```

2.2.3 Traditional 2-opt

The 2-opt [23-25] algorithm is a well-established local search heuristic extensively utilized to enhance the solutions for the TSP. This algorithm operates on the principle of iteratively swapping pairs of non-adjacent edges within the current solution. The primary aim of these swaps is to eliminate edge crossings that contribute to longer routes, thereby systematically reducing the overall length of the tour. This process of swapping is repeated until the algorithm reaches a state where no additional improvements in the tour length are possible, indicating a local optimum.

In practice, while the 2-opt algorithm does not ensure the discovery of the globally optimal solution, it is highly regarded for its ability to deliver solutions of commendable quality with considerable efficiency. The effectiveness of the 2-opt heuristic stems from its simplicity and the relatively low computational overhead involved in performing edge swaps, making it particularly suitable for moderately sized problem instances where a near-optimal solution is acceptable.

The workings of the 2-opt algorithm can be concisely described using the following pseudo code, which delineates the step-by-step process employed by the algorithm.

Algorithm 3: 2-opt Algorithm

```
pseudo tsp_2opt (distance matrix, num iterations=1000):
    num cities = get the number of cities from the distance matrix
    best path = list of cities in the order from 0 to num cities - 1
    best distance = calculate distance (best path, distance matrix)
    for _ in range (num iterations):
      improved = False
      for i from 1 to num cities - 2:
        for j from i + 1 to num cities:
          if j - i == 1:
            continue
          new path = copy of best path
          reverse the order of cities from index i to j in new path
```

```
            new distance = calculate (new path, distance matrix)
            if new distance < best distance:
                best path = new path
                best distance = new distance
                improved = True
        if not improved:
            break
    return best path, best distance
end pseudo
```

2.2.4 2-opt with Random Shuffle (2-optRS*)*

This study aimed to enhance the efficiency of the 2-opt algorithm within the generate an initial tour using a construction heuristic step. To achieve this objective, we introduced a random shuffle process, which entailed the random rearrangement of the elements in each iteration. This additional step was implemented to increase the likelihood of obtaining the shortest route.

For example, generate an initial tour step by the random shuffle.

Before random shuffle:

[1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19]

After random shuffle:

[3 11 1 9 14 2 4 15 19 6 8 12 7 18 16 13 10 5 17]

The pseudo-code of random shuffle can be explained as follows.

Algorithm 4: 2-optRS Algorithm

```
pseudo tsp_2optRS (distance matrix):
    initial tour = range (1, num cities)
    shuffle (initial tour)
    tour = concatenate ([0], initial tour) // starting city 0
    improved tour = tsp_2opt(tour) // Algorithm 3
end pseudo
```

The 2-opt algorithm, widely used for its efficiency in refining solutions to the TSP, shows enhanced performance when preceded by an initial random shuffle. This improvement can be attributed to several factors. Firstly, the random shuffle facilitates a broader exploration of the solution space, allowing the algorithm to avoid local minima and potentially discover more optimal routes. By starting with a varied initial solution, the algorithm is not confined to a narrow path that may lead to suboptimal results. Additionally, this randomized approach helps in mitigating biases inherent in the input data. Such biases might otherwise lead the algorithm to repeatedly favor certain patterns or sequences, thereby limiting its effectiveness. Lastly, the random shuffle disrupts fixed patterns of connections, which are often a result of sequential or structured data inputs. This disruption is crucial as it prevents the algorithm from settling into predictable loops and encourages the exploration of novel pathways that may yield shorter circuits. Overall, the integration of a random shuffle into the 2-opt algorithm not only diversifies the search process but also significantly enhances the possibility of finding a more efficient solution to the TSP [26].

2.3. Evaluation Metrics

This study utilized the Python programming language for the development and experimentation of the algorithms. The experiments were conducted by executing the code on a Colab Pro+ platform with a GPU V100. The evaluation metrics for assessing the performance of each algorithm can include the following.

2.3.1 Distance Measurement

To accurately measure the effectiveness of each algorithm in solving the TSP, we calculate the total travel distance of the tour generated by each algorithm [8, 27-28]. This calculation is performed by summing the distances between consecutive cities as per the tour sequence determined by the algorithm. These distances are derived from a pre-established matrix or direct measurements, ensuring that the data reflects realistic travel scenarios between locations.

For a comprehensive evaluation, each algorithm is subjected to multiple runs specifically, we execute each algorithm 10 times. This repetition is crucial for mitigating variations in performance due to stochastic elements or initial conditions in the algorithms, especially those incorporating randomization techniques such as the 2-optRS. By running each algorithm multiple times, we can more reliably assess their performance and ensure the robustness of our results.

After the runs, we identify and record the shortest total distance achieved in any of the cycles for each algorithm. This step is essential as it allows us to focus on the best possible outcome for each algorithm, providing a clear basis for comparison. We then compare these shortest distances across the different algorithms to determine their relative effectiveness. This comparison not only highlights which algorithm is most efficient in terms of minimizing travel distance but also sheds light on the potential trade-offs between the execution time and the optimality of the route. Selecting the tour with the shortest

distance from among multiple runs ensures that our evaluation is not biased by outlier results and reflects a truly effective solution to the TSP under the given conditions.

2.3.2 Runtime Performance

To thoroughly assess the performance of each algorithm in solving the TSP, we adopted a methodical approach by focusing on the execution time as a key performance metric. Specifically, we executed each algorithm through a series of 10 iterations to ensure robustness in our measurements, capturing the runtime for each iteration [27-28]. This repetitive testing is critical as it helps to mitigate any anomalies due to transient system conditions or external interferences that might affect computational performance. After collecting runtime data from all ten trials, we computed the average execution time for each algorithm. This average provides a reliable measure of the algorithm's efficiency under controlled conditions and allows for a fair comparison across different algorithms.

This quantitative method of evaluation using execution times enables us to systematically compare the computational efficiency of each algorithm. By analyzing these times, we can identify which algorithm achieves its goal with minimal processing time, thereby offering insights into its suitability for real time applications or scenarios where speed is paramount. Furthermore, this approach helps in understanding the trade-offs between the quality of the solution provided by each algorithm and the time it requires to reach that solution, thus guiding users in choosing the appropriate algorithm based on their specific performance and efficiency needs.

3. RESULTS AND DISCUSSION

This section offers a detailed and systematic analysis of the experimental outcomes derived from the implementation of the proposed algorithms, with an emphasis on evaluating their performance relative to conventional algorithms. The research was conducted using geographical data from the northeastern region of Thailand, encompassing the central locations of 20 provinces. The actual distances between these provinces were meticulously sourced from Google Maps to ensure accuracy in route measurements, and the detailed results of these measurements are systematically presented in Table 1.

To maintain consistency and comparability in our analysis, the city of Kalasin was designated as the starting point for all algorithms across each iteration of the experiments. This consistent starting point ensures that the comparative analysis of the algorithmic performance is fair and controlled, allowing for a direct evaluation of the efficiency and effectiveness of each algorithm without confounding variables.

The core experimental results concerning the traditional 2-opt algorithm and the newly proposed 2-optRS algorithm are encapsulated in Table 1. This table serves as a crucial comparative tool, illustrating both the total distance traversed by each algorithm and the corresponding computational time required for each run. Specifically, the table delineates the comparative performance of the 2-opt and 2-optRS algorithms, providing a detailed breakdown of processing times and the total distances calculated. By presenting these metrics, the table enables a clear and direct comparison between the two algorithms, highlighting differences in efficiency and effectiveness in solving the TSP under the specified conditions of the study. Through this comparative analysis, we aim to illustrate the potential improvements in route optimization and computational efficiency introduced by the 2-optRS algorithm.

Table 1. Comparative Performance of 2-opt and 2-optRS Algorithms

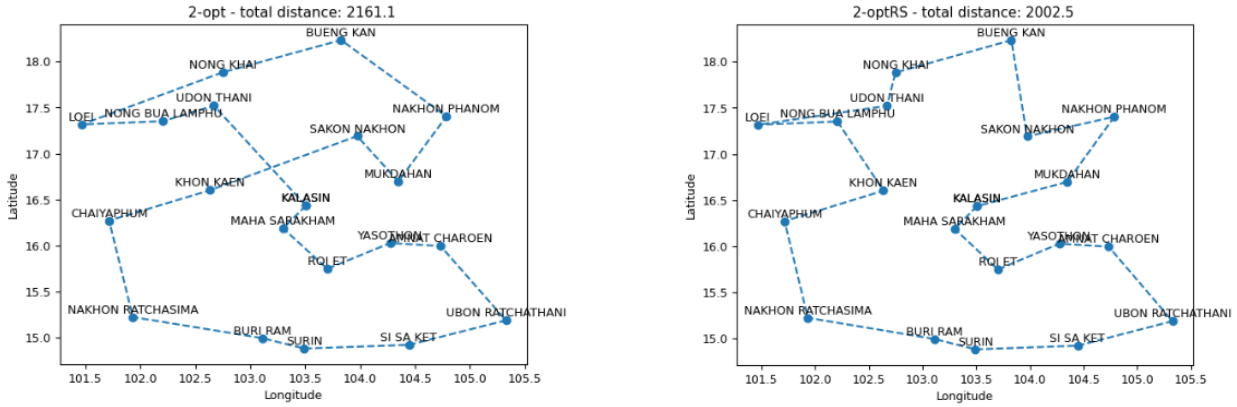
Times	2-opt		2-optRS	
	Total distance (km)	Runtime (sec)	Total distance (km)	Runtime (sec)
1	2161.1	0.00358	2509.5	0.00826
2	2161.1	0.00359	2225.9	0.00680
3	2161.1	0.00354	2174.1	0.00951
4	2161.1	0.00356	2363.5	0.00637
5	2161.1	0.00336	2090.0	0.00732
6	2161.1	0.00360	2114.6	0.00761
7	2161.1	0.00532	**2002.5**	0.00871
8	2161.1	0.00363	2068.8	0.01001
9	2161.1	0.00351	2344.2	0.00948
10	2161.1	0.00358	2213.5	0.01056

Table 1 provides a systematic comparison of the 2-opt and 2-optRS algorithms based on experimental results derived from this study. It is evident from the data that the integration of a random shuffle technique into the conventional 2-opt algorithm, denoted as 2-optRS, significantly enhances its capability to minimize the travel distance in the TSP. Specifically, the minimum distance achieved by the 2-optRS algorithm was 2002.5 km, which presents a substantial improvement, reducing the travel distance by 158.6 km compared to the traditional 2-opt algorithm, which consistently produced a distance of 2161.1 km across all runs.

However, this enhancement in route optimization comes with a trade-off in computational time. The data indicates that the runtime required for the 2-optRS algorithm is consistently longer than that of the standard 2-opt. This increase in

runtime highlights the computational cost associated with implementing the random shuffle procedure, which, while effective in reducing the travel distance, demands more processing resources.

Figure 3 illustrates these findings visually, with part (a) displaying the path results obtained from the standard 2-opt algorithm and part (b) showing the optimized path generated by the 2-optRS algorithm. This visual representation complements the quantitative data presented in Table 1, providing a clear depiction of the practical impact of the algorithmic enhancements on the TSP solutions.



a. b.

Figure 3. The path result: (a) 2-opt (b) 2-optRS (Our)

Figure 3 provides a visual comparison of the routes generated by the 2-opt and 2-optRS algorithms, offering a clear illustration of their respective performance in solving the TSP in the northeastern region of Thailand. This figure comprises two sub-figures: part (a) showing the path from the traditional 2-opt algorithm with a total distance of 2161.1 km, and part (b) depicting the optimized path achieved by the 2-optRS algorithm, which recorded a shorter distance of 2002.5 km.

From the visual representation, it is evident that the path optimization achieved by the 2-optRS algorithm not only reduces the total travel distance but also alters the travel sequence between provinces. This observation suggests that the introduction of a random shuffle technique effectively diversifies the initial route configurations, thereby enabling the algorithm to explore more varied paths that potentially bypass longer routes inherent in the initial setup used by the traditional 2-opt algorithm. The path generated by the 2-optRS algorithm (Figure 3b) demonstrates fewer long-distance connections between provinces, which is indicative of a more efficient routing logic. In contrast, the 2-opt path (Figure 3a) includes several extended travel links, which contribute to its overall longer circuit. This difference highlights the 2-optRS algorithm's ability to break away from less optimal initial solutions and explore a broader solution space that likely contributes to finding shorter and more efficient routes.

Moreover, the comparative visual analysis also underscores the computational trade-off involved. While the 2-optRS algorithm achieves a shorter path, it does so at the cost of increased runtime, as previously noted in Table 1. This trade-off is a crucial consideration for practical applications where both route efficiency and computational resources are critical factors. In conclusion, Figure 3 not only corroborates the quantitative data presented in Table 1 but also provides a graphical elucidation of how the random shuffle mechanism enhances the performance of the 2-opt algorithm. The implications of these findings are significant for optimizing logistic routes, reducing transportation costs, and improving overall operational efficiency in geographical routing problems.

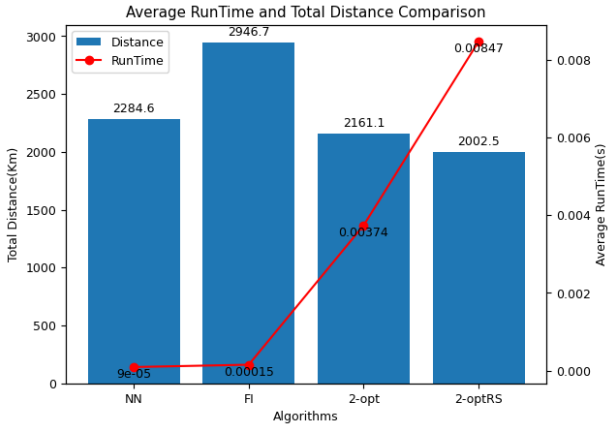


Figure 4: The comparison results total distance and runtime.

In this study, we conducted an extensive comparative analysis of various algorithms, including the newly developed 2-optRS, the traditional 2-opt, Nearest Neighbor (NN), and Farthest Insertion (FI) algorithms, focusing on their effectiveness in solving the TSP. The primary metrics for this comparison were the total travel distance achieved by each algorithm and their computational runtimes, as depicted in Figure 4.

The data from Figure 4 illustrates that the 2-optRS algorithm substantially outperforms the Nearest Neighbor and Farthest Insertion algorithms in terms of minimizing the total travel distance. Specifically, the 2-optRS algorithm achieved a total travel distance of 2002.5 km, significantly less than the 2284.6 km and 2946.7 km distances achieved by the Nearest Neighbor and Farthest Insertion algorithms, respectively. This represents a reduction in travel distance of 282.1 km compared to NN and 944.2 km compared to FI. This improvement highlights the efficacy of the 2-optRS algorithm in optimizing route efficiency, leveraging the random shuffle technique to explore more effective pathways that reduce the overall travel distance.

However, the advancements in distance minimization come with a notable increase in computational time. As shown in the figure, while the 2-optRS algorithm reduces the travel distance substantially, it also requires a longer processing time marked at approximately 0.00847 seconds on average compared to 0.00015 seconds and 9e-05 seconds for the FI and NN algorithms, respectively. This increase in runtime is attributed to the more complex computations involved in the random shuffle and subsequent optimization steps in the 2-optRS algorithm.

The comparative analysis [29] underscores a critical trade-off between minimizing travel distance and optimizing computational efficiency. While the 2-optRS algorithm offers significant improvements in route optimization, the increased runtime could be a limiting factor in scenarios where both time efficiency and distance minimization are crucial.

In conclusion, the results from Figure 4 provide a comprehensive overview of the strengths and limitations of the 2-optRS algorithm compared to traditional TSP-solving methods. It highlights the potential of advanced optimization techniques to significantly enhance route efficiency at the expense of increased computational demand. This balanced evaluation is crucial for practical applications, where the choice of algorithm may depend on specific operational requirements and constraints. The findings serve as a valuable benchmark for future research and applications in route optimization within logistic and transportation planning domains.

4. CONCLUSION

This research focused on investigating the TSP and aimed to compare the performance of various heuristic algorithms, including nearest neighbor, farthest insertion, the traditional 2-opt algorithm, and the author's proposed algorithm that incorporates an enhancement to the 2-opt algorithm by introducing a random shuffle (2-optRS). The study collected new real route information from Google Maps, specifically from the northeastern region of Thailand, encompassing a total of 20 provinces. The dataset used in this study is made available for future research and interested parties [30]. Through extensive experimentation, the results demonstrated that the proposed algorithm outperformed all other algorithms by consistently yielding shorter total distances. This finding underscores the effectiveness of the proposed method in achieving improved solutions to the TSP. The research findings contribute to the existing body of knowledge on TSP-solving algorithms and offer potential insights for practical applications in optimizing travel routes and logistics planning. Further investigations can be conducted to explore the algorithm's performance on larger datasets or to examine its adaptability to different geographical regions or problem instances.

REFERENCES

[1] M. M. Flood, "The traveling-salesman problem," *Operations research*, vol. 4, no. 1, pp. 61-75, 1956.
[2] G. A. Croes, "A method for solving traveling-salesman problems," *Operations research*, vol. 6, no. 6, pp. 791-812, 1958.
[3] M. Jünger, G. Reinelt, and G. Rinaldi, "The traveling salesman problem," *Handbooks in operations research and management science*, vol. 7, pp. 225-330, 1995.
[4] K. L. Hoffman, M. Padberg, and G. Rinaldi, "Traveling salesman problem," *Encyclopedia of operations research and management science*, vol. 1, pp. 1573-1578, 2013.
[5] R. Kumar and M. Memoria, "A review of memetic algorithm and its application in traveling salesman problem," *International Journal on Emerging Technologies*., vol. 11, no. 2, pp. 1110-1115, 2020.
[6] L. Zambito, "The traveling salesman problem: a comprehensive survey," *Project for CSE*, vol. 4080, 2006.
[7] Ali, Z. A., Rasheed, S. A., & Ali, N. N. M.. An enhanced hybrid genetic algorithm for solving traveling salesman problem. *Indonesia Journal of Electrical Engineering and Computuper Science*, 18(2), 1035-1039, 2020, doi: 10.11591/ijeecs.v18.i2.pp1035-1039
[8] O. Abdoun and J. Abouchabaka, "A comparative study of adaptive crossover operators for genetic algorithms to resolve the traveling salesman problem," *arXiv preprint arXiv*:1203.3097, 2011.
[9] H. Zhou, M. Song, and W. Pedrycz, "A comparative study of improved GA and PSO in solving multiple traveling salesmen problem," *Applied Soft Computing*, vol. 64, pp. 564-580, 2018, doi:10.1016/j.asoc.2017.12.031
[10] Department of Public Works and Town & Country Planning, "Executive Report: Analysis of the area situation Northeast Thailand," 2020. [Online]. Available: http://subsites.dpt.go.th/edocument/

[11] Anggara, D, "Decision Support System SAW Method Exporter Foreign Trade Section". *Journal of Information Systems and Technology Research*, 1(1), 23-31, 2022. doi: 10.55537/jistr.v1i1.91
[12] 'Google Maps Platform', Google for Developers. [Online]. Available: https://developers.google.com/maps
[13] Onsurathum, S., Pinlaor, P., Charoensuk, L., Haonon, O., Chaidee, A., Intuyod, K., & Pinlaor, S. Contamination of Opisthorchis viverrini and Haplorchis taichui metacercariae in fermented fish products in northeastern Thailand markets. Food Control, 59, 493-498. 2016, doi: 10.1016/j.foodcont.2015.06.020.
[14] D. R. Lanning, G. K. Harrell, and J. Wang, "Dijkstra's algorithm and Google maps," *Proceedings of the 2014 ACM Southeast Regional Conference*, 2014, pp. 1-3.
[15] A. Javaid, "Understanding Dijkstra's algorithm," Available at SSRN 2340905, 2013.
[16] B. Golden, L. Bodin, T. Doyle, and W. Stewart Jr, "Approximate traveling salesman algorithms," *Operations research*, vol. 28, no. 3-part-ii, pp. 694-711, 1980.
[17] S. Desale, A. Rasool, S. Andhale, and P. Rane, "Heuristic and meta-heuristic algorithms and their relevance to the real world: a survey," *International Journal of Computer Engineering in Research Trends*, vol. 351, no. 5, pp. 2349-7084, 2015.
[18] S. Gupta, A. Rana, and V. Kansal, "Comparison of Heuristic techniques: A case of TSP," in 2020 *10th International Conference on Cloud Computing, Data Science & Engineering (Confluence)*, 2020: IEEE, pp. 172-177.
[19] G. Kizilateş and F. Nuriyeva, "On the nearest neighbor algorithms for the traveling salesman problem," *Proceedings of the Third International Conference on Computational Science, Engineering and Information Technology*, KTO Karatay University, June 7-9, 2013, Konya, Turkey-Volume 1, 2013: Springer, pp. 111-118.
[20] M. A. Rahman and H. Parvez, "Repetitive nearest neighbor based simulated annealing search optimization algorithm for traveling salesman problem," *Open Access Library Journal*, vol. 8, no. 6, pp. 1-17, 2021.
[21] M. Sahin, "Solving TSP by using combinatorial Bees algorithm with nearest neighbor method," *Neural Computing and Applications*, vol. 35, no. 2, pp. 1863-1879, 2023. doi: 10.1007/s00521-022-07816-y
[22] S. Poikonen, B. Golden, and E. A. Wasil, "A branch-and-bound approach to the traveling salesman problem with a drone," *INFORMS Journal on Computing*, vol. 31, no. 2, pp. 335-346, 2019, doi: 10.1287/ijoc.2018.0826
[23] M. Hasegawa, T. Ikeguchi, and K. Aihara, "Combination of chaotic neurodynamics with the 2-opt algorithm to solve traveling salesman problems," *Physical Review Letters*, vol. 79, no. 12, p. 2344, 1997.
[24] M. Englert, H. Röglin, and B. Vöcking, "Worst Case and Probabilistic Analysis of the 2-Opt Algorithm for the TSP," *Algorithmica*, vol. 68, no. 1, pp. 190-264, 2014/01/01 2014, doi: 10.1007/s00453-013-9801-4.
[25] P. R. d O Costa, J. Rhuggenaath, Y. Zhang, and A. Akcay, "Learning 2-opt heuristics for the traveling salesman problem via deep reinforcement learning," in *Asian Conference on Machine Learning*, 2020: PMLR, pp. 465-480.
[26] Uddin, F., Riaz, N., Manan, A., Mahmood, I., Song, O. Y., Malik, A. J., & Abbasi, A. A.. An Improvement to the 2-Opt Heuristic Algorithm for Approximation of Optimal TSP Tour. *Applied Sciences*, 13(12), 7339. 2023, doi: 10.3390/app13127339.
[27] J. Wang, C. Xiao, S. Wang, and Y. Ruan, "Reinforcement Learning for the Traveling Salesman Problem: Performance Comparison of Three Algorithms," *Authorea*. May 12, 2023, doi: 10.22541/au.168389655.51789479/v1
[28] R. T. Bye, M. Gribbestad, R. Chandra, and O. L. Osen, "A comparison of ga crossover and mutation methods for the traveling salesman problem," in Innovations in Computational Intelligence and Computer Vision: *Proceedings of ICICV 2020, 2021: Springer*, pp. 529-542.
[29] Liu, X. X., Liu, D., Yang, Q., Liu, X. F., Yu, W. J., & Zhang, J. "Comparative analysis of five local search operators on visiting constrained multiple traveling salesmen problem". *In 2021 IEEE Symposium Series on Computational Intelligence (SSCI)* (pp. 01-08). IEEE. 2021. doi: 10.1109/SSCI50451.2021.9659963
[30] Kamthorn sarawan, 'Distance of Provinces in Northeast Thailand' [Online]. Available: https://github.com/kamthornsa/provinces-northeast-thailand