Solving Traveling Salesman Problem Art Using Clustered Traveling Salesman Problem

Nadya Sulistia, Irwansyah, and Marwan

***Abstract*—The Traveling Salesman Problem (TSP) is a well-known optimization problem that seeks to determine the shortest possible route that allows a salesman to visit each city exactly once before returning to the starting point. With advances in TSP theory and its applications, a novel concept known as TSP Art has emerged, blending mathematics with artistic expression. In TSP Art, the optimal solution to the TSP generates an artistic pattern or figure. However, the complexity of this problem increases with the number of vertices, making it computationally challenging to solve. This study proposes an approach using the Clustered Traveling Salesman Problem (CTSP) to address the TSP Art problem by organizing vertices into clusters, where each cluster is visited once, while maintaining an efficient overall tour. The objective of this research is to solve the TSP Art problem using the CTSP approach and to calculate the length of the minimum tours. The Nearest Neighbor and 2-opt algorithms are applied within each cluster to find the shortest paths, while Kruskal's algorithm is employed to connect these paths into an optimized overall tour. The minimum tour lengths for TSP Art representations of Mona Lisa, Van Gogh, and Venus are determined to be** $6,932,014.19, 6,878,519.41,$ **and** $8,210,589.60$ **distance unit, respectively.**

***Index Terms*—Clustered Traveling Salesman Problem, Kruskal's Algorithm, Nearest Neighbor Algorithm, TSP Art.**

I. Introduction

THE traveling salesman problem (TSP) is a method for finding the minimum tour in a graph. A graph G consists of vertex set $V(G)$ and edge set $E(G)$, and the vertices form ordered pairs called edges. In the TSP, every edge has a cost or distance associated with it [1]. The goal is to find the minimum tour that satisfies the condition that each vertex is visited exactly once before returning to the starting point. As TSP theory developed, TSP art emerged as a mathematical concept applied to art, where vertices and edges form artistic figures. The large number of vertices in TSP art makes finding a solution more challenging. There are many TSP art data could be access today. This research is using Mona Lisa, Van Gogh, and Venus TSP art data from http://www.math.uwaterloo.ca/tsp/data/art/.

According to Johnson and McGeoch [2], there are two main methods for solving large-scale Traveling Salesman Problem (TSP) instances. The first method is the Lin-Kernighan (LK) algorithm, which focuses on local optimization and has also been implemented in various modified versions such as Chained LK and LKH-2. The LKH-2 is particularly effective for finding high-quality tours by iteratively improving solutions through local adjustments. The second method uses Genetic Algorithms (GA), specifically hybrid Memetic Algorithms (MA), which combine GA with the LK algorithm. Some versions of these methods, such as GA with Edge Assembly Crossover (GA-EAX), have shown excellent performance, even for TSP instances with up to 200,000 cities.

Another successful study addressing large-scale TSP problems with many cities is Honda's research, which uses a Parallel Genetic Algorithm with Edge Assembly Crossover (Parallel GA-EAX). Edge Assembly Crossover (EAX) is a method designed to generate new solutions by swapping edges between two parent tours, while also maintaining diversity within the population. The use of a Parallel Genetic Algorithm efficiently distributes tasks across multiple processors using a master/worker model, significantly reducing computational time without sacrificing accuracy. The algorithm follows a Two-Stage Approach, the first phase focuses on local edge exchanges to explore diverse solutions, while the second phase involves global edge exchanges to further improve the best solutions found. This research significantly reduces the running time compared to previous studies by using a parallel framework that distributes the workload across many processors and employs a specialized crossover method (EAX) to maintain solution diversity and improve quality in two stages [3].

This research, however, uses the clustered traveling salesman problem (CTSP). The CTSP is a modification of TSP that clusters vertices into groups, with each cluster need to be visited exactly once [4]. In large-scale TSP problems, handling all cities simultaneously can consume massive amounts of memory (RAM). Clustering breaks the problem into smaller groups. By grouping cities into clusters, each cluster is solved individually, reducing the overall memory requirement. Furthermore, clustering is computationally more efficient than handling the entire dataset at once. This speeds up the algorithm and allows it to handle large-scale problems more effectively [5]. Moreover, the nearest neighbor and 2-opt algorithms are used to find the shortest path in each cluster, and Kruskal's algorithm is employed to connect these paths to form the minimum tour.

The nearest neighbor (NN) algorithm is a straightforward and fast greedy approach. It begins at a city and continues by selecting the closest unvisited city. Although it may not find the best possible solution, it can quickly create a path, which is helpful for large problems with many vertices, where finding the perfect solution would take too much time [6]. The 2-opt algorithm swaps vertices in order to improve the optimization result. Meanwhile, Kruskal's algorithm gradually constructs

The authors are with the Department of Mathematics, University of Mataram, Indonesia, e-mail: sulistianadya@gmail.com, irw@unram.ac.id, marwan.math@unram.ac.id

Manuscript received March 31, 2024; accepted September 25, 2025.

the solution by sorting the edges by cost. Additionally, the characteristics of Kruskal' algorithm simplify and improve the formation of the path [7].

II. Clustered Traveling Salesman Problem

The traveling salesman problem (TSP) involves finding a minimum tour that visits N cities, starting from one city and returning to the same city, with each city visited exactly once before returning. Additionally, the distance of the tour must be as short as possible. In the TSP, these cities and roads are represented as a graph, where each city is a vertex and a road between two cities is an edge. Each edge has a cost. The TSP model is defined as follows [8]

$$\min Z = \sum_{i=1}^{N}\sum_{j=1}^{N} c_{ij}a_{ij} \tag{1}$$

with constraints

$$\sum_{i=1}^{N} a_{ij} = 1, j = 1, 2, ..., N \tag{2}$$

$$\sum_{j=1}^{N} a_{ij} = 1, i = 1, 2, ..., N \tag{3}$$

where

$$a_{ij} = \begin{cases} 1, & \text{if (i,j) is chosen to be part of the tour} \\ 0, & \text{others} \end{cases} \tag{4}$$

and c_{ij} is the cost of an edge between vertices i and j. In order to prevent a loop (an edge connecting a vertex to itself), a large constant M is assigned to any edge where $i = j$:

$$c_{ij} = M, \forall i = j \tag{5}$$

Here, i and j are index for the vertices and N is the number of vertices. Furthermore, TSP art is a variant of TSP where the minimum tour forms an object or a painting [9].

The large number of vertices in the TSP art problem makes finding a solution more complicated and increases the running time. To address these issues, the clustered traveling salesman problem (CTSP) is used in this research. CTSP is a variation of the TSP where vertices are grouped into clusters, and each cluster must be visited exactly once [4].

Before determining the minimum path, the data should be clustered. This research uses the K-Means clustering method, which groups vertices based on their distance to the centroid of a cluster. A vertex is assigned to the cluster with the smallest distance to its centroid [10], [11]. An optimum number of clusters is determined by using the elbow method. In the elbow method, the optimal number of clusters is determined by analyzing a graph that shows the ratio of K (number of clusters) to its distortion (Mean Squared Error) value. The K value is selected based on the position of the "elbow" in the graph [11], [12].

Given the set of vertices $V(G)$, with N total vertices and K clusters, define $V = \{v_1, v_2, \ldots, v_N\}$ and $C = \{C_1, C_2, \ldots, C_K\}$, where C_k represents k-th cluster, $k \in 1, 2, \ldots, K$, and $K \leq N$. Let μ_k be the centroid of C_k, and $\mu_k = (\mu_k(x), \mu_k(y))$, then

$$\mu_k(x) = \frac{1}{|C_k|}\sum_{x_j \in C_k} V_j \tag{6}$$

$$\mu_k(y) = \frac{1}{|C_k|}\sum_{y_j \in C_k} V_j \tag{7}$$

After the data clustered into K clusters, the distance between each pair of vertices (c_{ij}) is calculated using the Euclidean distance formula. Given vertices $v_i = (x_i, y_i)$ and $v_j = (x_j, y_j)$, we have

$$c_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \tag{8}$$

These distances are then used to determine the minimum path within each cluster using the nearest neighbor and 2-opt algorithms. Once the minimum path for each cluster is found, the next step is to connect all these paths into a minimum tour. The connection must link different clusters, and cycles must be avoided to ensure no vertex is visited more than once, as this would violate the conditions of a minimum tour. To prevent cycles, the start and end points of the minimum path in each cluster are assigned a large distance to avoid being chosen as connectors. The connectors between clusters are determined using Kruskal's algorithm, ensuring all vertices are finally connected.

The steps for solving TSP art in this research are as follows

- Cluster the data into K clusters using K-Means Clustering [13] [12].
  1) Determine the number of clusters (K).
  2) Define the initial positions of the centroids.
  3) Calculate the distance between each vertex and the centroids using the Euclidean distance.
  4) Assign each vertex to the cluster with the smallest distance to its centroid.
  5) Update the centroid position by calculating the average distance of each cluster's members.
  6) Repeat step 3 through 5.
  7) Stop when the centroid positions no longer change.
- Determine the optimal number of clusters (K) using elbow method [14].
  1) Start with $K = 1$, determine the centroid positions, and calculate the distortion value.
  $$distortion = \frac{1}{N}\sum_{k=1}^{K}\sum_{i=1}^{|C_k|}(V_i - \mu_k)^2 \tag{9}$$
  2) Increase K to $K + 1$ and repeat the clustering process.
  3) Using the K-Means clustering, the vertices must be clustered into K clusters.
  4) Calculate the new distortion.
  5) Repeat steps (2)-(4)
  6) Plot the number of clusters (K) on the x-axis and distortion on the y-axis.
  7) Choose the K value at the elbow point of the graph.

- Cluster the data into the optimal number of clusters found in the previous step using K-Means clustering.
- Calculate the cost of each vertex pair within the same cluster using Euclidean distance.
- Determine the minimum path for each cluster using the nearest neighbor and 2-opt algorithms.
- Each minimum path of every cluster has an initial and an end point, which will be connected to the initial or end point of another cluster. In this step, the connector is determined using Kruskal's algorithm.
- We have found a good solution based on the nearest neighbor, 2-opt, and Kruskal's algorithms.

III. Results and Discussions

Before determining the minimum path, the data should be clustered. The Figure 1 shows the plot of number of clusters versus its corresponding distortion values.

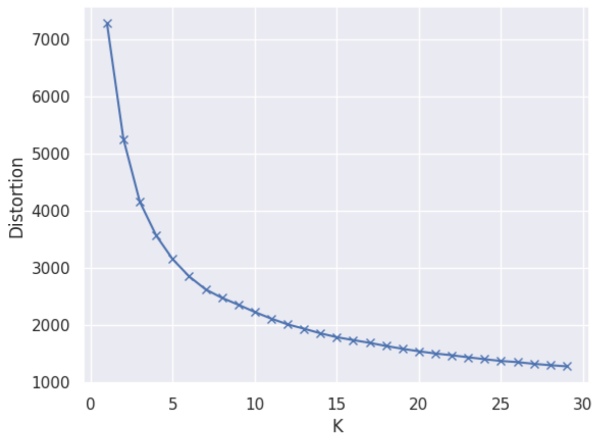

Fig. 1: Number of clusters versus distortion values

The figure indicates that the optimal number of clusters (K) for Mona Lisa's data is 7. The Figure 2 shows the data after clustering. There are 7 black points representing the centroid of each cluster. Each color represents one cluster.

After clustering the data, the minimum path for each cluster is determined using the nearest neighbor and 2-opt algorithms. Before calculating the minimum path, the cost of each edge is determined using the Euclidean distance (see equation 8). The cost of each edge is stored in a matrix called the distance matrix. The diagonal elements of the matrix are given a large constant, M, to prevent them from being chosen, which avoids loops (c_{ij} where $i = j$).

The starting point is chosen based on the first vertex in the clustered data, denoted as v_0. From the starting point, the next vertex to visit is chosen based on the minimum cost (c_{ij}) among all edges adjacent to the starting point. This is done by examining row v_0. Base on that row in the distance matrix. The minimum cost is found for a specific vertex, v_d, making c_{0d} the minimum cost, and edge e_{0d} is the first edge in the minimum path. Once e_{0d} is selected, vertices v_0 and v_d should not be visited again. To prevent this, the columns corresponding to v_0 and v_d are assigned (M). The same process is repeated for vertex v_d, finding the adjacent vertex with the next minimum

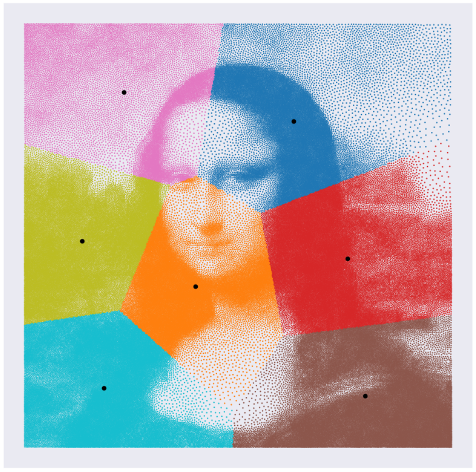

Fig. 2: Clustered data

cost, $v_{d'}$. Thus, edge $e_{dd'}$ becomes part of the minimum path. Again, the columns for v_d and $v_{d'}$ are assigned (M). This process continues until $|C_k| - 1$ edges are found to form the minimum path for each cluster, where $|C_k|$ represents number of elements in cluster C_k. The 2-opt algorithm used to improve the results from nearest neighbor heuristic. The result of the calculation is shown in Table I.

TABLE I: Minimum distance in each cluster

Cluster (k)	Minimum distance
0	$942,576.76$
1	$1,019,512.67$
2	$744,869.05$
3	$766,935.38$
4	$945,455.40$
5	$885,896.206$
6	$700,853.47$

The Figure 3 shows the initial and end points of the minimum path for each cluster, represented by black points.

Those points are used to connect one cluster to another, forming a minimum tour. Each vertex is assigned an index. Indices 0 to 6 represent the initial points for clusters 0 through 6, respectively, while indices 7 to 13 represent the end points for clusters 0 through 6, respectively.

In order to select the connector edge for each cluster, Kruskal's algorithm is applied. Two conditions must be satisfied when choosing these connectors. First, the cost of a loop must be a large constant, denoted as M. Second, the initial and end points of the same cluster cannot be connected, as this would create a local cycle in a cluster. To avoid this, assign M as the cost for the initial and end points of the same cluster as follows:

$$w_{ij} = \begin{cases} M, & \text{for } |i-j| \mod K = 0 \\ c_{ij}, & \text{others} \end{cases} \tag{10}$$

Here, w_{ij} is the cost of a connector or edge consisting of vertices v_i and v_j), where i and j are indices ranging from 1

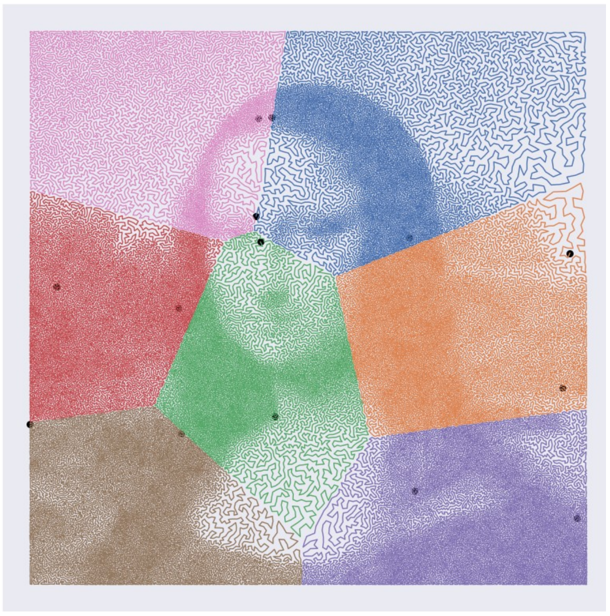

Fig. 3: Minimum path in each cluster with its initial and end points

to U, and U is the number of end points.

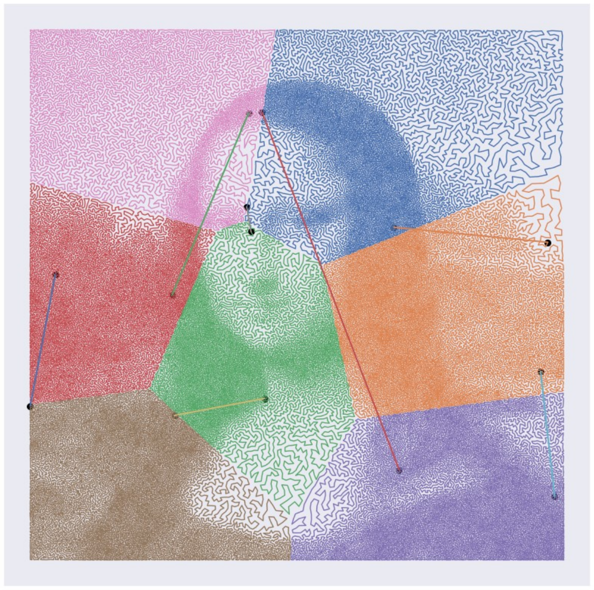

Fig. 4: Clusters connection

The cost of each connector can be seen in Table II. The total

TABLE II: Connector's cost

Connector	Connector's Cost
0	941.862517
1	$3,436.688086$
2	$4,748.446588$
3	$5,049.083283$
4	$5,791.527605$
5	$7,422.693649$
6	$14,408.554993$

cost of these connectors is $41,798.85672072333$ unit distance. The total minimum path cost for all clusters is $6,006,098.93$ unit distance. Therefore, the minimum tour is $6,047,897.79$ unit distance (for the tour in Figure 4).

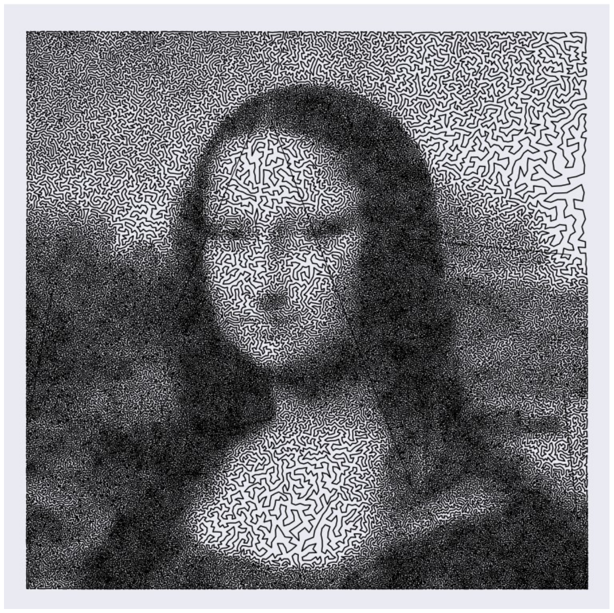

Fig. 5: Final result

In Figure 5, the black points marking the initial and end points were removed, and the color of each cluster was changed to the same color to show the final result of the Mona Lisa TSP art.

The same steps are applied to calculate the minimum tour for the Van Gogh and Venus datasets, yielding distances recorded in Table III. Moreover, Table III shows the comparison between our results and current best results recorded in https://www.math.uwaterloo.ca/tsp/data/art/

TABLE III: Comparison to current best tour

Datasets	Our results	Current best
Mona Lisa	6,047,897.79	5,757,191
Van Gogh	6,878,519.41	6,543,609
Venus	7,138,938.72	6,810,665

IV. Conclusion

To solve TSP art using CTSP, this research suggests clustering all vertices with the K-Means algorithm and determining the optimal number of clusters using the Elbow method. Each vertex is grouped accordingly, and the minimum path within each cluster is calculated using the nearest neighbor algorithm. Finally, the initial and end points of these paths are connected using Kruskal's algorithm, and thus, the minimum tour is determined.

Acknowledgment

The authors would like to thank the anonymous reviewer for their valuable suggestions and insightful comments, which have significantly improved the quality of this work.

References

[1] J. A. Bondy and U. S. R. Murty, *Graph theory*. Springer Publishing Company, Incorporated, 2008.

[2] D. S. Johnson and L. A. McGeoch, "The traveling salesman problem: a case study," *Local search in combinatorial optimization*, pp. 215–310, 1997.

[3] K. Honda, Y. Nagata, and I. Ono, "A parallel genetic algorithm with edge assembly crossover for 100,000-city scale tsps," in *2013 IEEE congress on evolutionary computation*. IEEE, 2013, pp. 1278–1285.

[4] Y. Lu, J.-K. Hao, and Q. Wu, "Solving the clustered traveling salesman problem via tsp methods," *arXiv preprint arXiv:2007.05254*, 2020.

[5] H. Xu and H. Lan, "An adaptive layered clustering framework with improved genetic algorithm for solving large-scale traveling salesman problems," *Electronics*, vol. 12, no. 7, p. 1681, 2023.

[6] B. A. AlSalibi, M. B. Jelodar, and I. Venkat, "A comparative study between the nearest neighbor and genetic algorithms: A revisit to the traveling salesman problem," *International Journal of Computer Science and Electronics Engineering (IJCSEE)*, vol. 1, no. 1, pp. 110–123, 2013.

[7] K. Salahddine *et al.*, "The implementation of kruskal's algorithm for minimum spanning tree in a graph," in *E3S Web of Conferences*, vol. 297. EDP Sciences, 2021, p. 01062.

[8] H. A. Taha and H. A. Taha, *Operations research: an introduction*. Prentice hall Upper Saddle River, NJ, 2003, vol. 7.

[9] R. Bosch and A. Herman, "Continuous line drawings via the traveling salesman problem," *Operations research letters*, vol. 32, no. 4, pp. 302–303, 2004.

[10] A. Agrawal and H. Gupta, "Global k-means (gkm) clustering algorithm: a survey," *International journal of computer applications*, vol. 79, no. 2, 2013.

[11] D. Sharifrazi, R. Alizadehsani, J. H. Joloudari, S. S. Band, S. Hussain, Z. A. Sani, F. Hasanzadeh, A. Shoeibi, A. Dehzangi, M. Sookhak *et al.*, "Cnn-kcl: Automatic myocarditis diagnosis using convolutional neural network combined with k-means clustering," *Mathematical Biosciences and Engineering*, vol. 19, no. 3, pp. 2381–2402, 2022.

[12] C. Shi, B. Wei, S. Wei, W. Wang, H. Liu, and J. Liu, "A quantitative discriminant method of elbow point for the optimal number of clusters in clustering algorithm," *Eurasip Journal on Wireless Communications and Networking*, vol. 2021, pp. 1–16, 2021.

[13] A. Sucipto, "Klasterisasi calon mahasiswa baru menggunakan algoritma k-means," *Science Tech: Jurnal Ilmu Pengetahuan dan Teknologi*, vol. 5, no. 2, pp. 50–56, 2019.

[14] P. Bholowalia and A. Kumar, "Ebk-means: A clustering technique based on elbow method and k-means in wsn," *International Journal of Computer Applications*, vol. 105, no. 9, 2014.