Comparative Analysis of Dynamic Programming, Genetic Algorithms, and Google OR-Tools for Optimizing kWh Meter Replacement Routes

Ike Ayu Idiara - 23522306
Program Studi Magister Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail (gmail): ikeidiara89@gmail.com

***Abstract*— In June 2023, PLN Unit Layanan Pelanggan (ULP) Tegalrejo in Magelang City, Central Java, faced a significant increase in kWh meter installation requests due to the implementation of the Advanced Metering Infrastructure (AMI) program. To meet the target of replacing kWh meters for 52,071 customers within six months, ULP Tegalrejo must optimize the replacement routes, addressing the limited number of available technicians. This study models the optimization problem as a Traveling Salesman Problem (TSP) and compares three methods: Dynamic Programming (DP), 2 approach of Genetic Algorithms (GA), and Google OR-Tools. The results demonstrate that Google OR-Tools consistently provides the most efficient and effective solutions, delivering the shortest distances and fastest execution times. The second approach of GA, utilizing tournament selection, single-point crossover, and elitism, also shows superior performance compared to the first approach. The DP method is effective for small-scale problems but becomes impractical for larger datasets due to high computational complexity.**

Keywords—Traveling Salesman Problem, Dynamic Programming, Google OR Tools, Genetic Algorithm

I. INTRODUCTION (*HEADING 1*)

In June 2023, PLN Unit Layanan Pelanggan (ULP) Tegalrejo in Magelang City, Central Java, faced a significant increase in kWh meter installation requests due to the implementation of the Advanced Metering Infrastructure (AMI) program. PLN aims to have 1,217,256 customers using AMI services by the end of 2023, with ULP Tegalrejo targeted to replace kWh meters for 52,071 customers within six months. To achieve this goal, ULP Tegalrejo must increase the number of Measuring and Limiting Device (APP) replacements from 30-50 to 320-340 jobs per day. However, the main challenge lies in the limited number of technicians available, which needs to be addressed to ensure the APP replacement and installation process meets the target.

To address this challenge, route optimization using the Traveling Salesman Problem (TSP) concept is essential. TSP involves finding the shortest possible route to visit all locations once and return to the starting point. Typically, TSP is solved using the brute force algorithm to guarantee the absolute best route. However, brute force becomes highly inefficient when the number of locations is very large, as in the case of ULP Tegalrejo with 52,071 customers/points to visit. The computational complexity of brute force methods grows factorially with the number of locations, making it impractical for large-scale problems. Consequently, alternative optimization techniques such as Genetic Algorithms (GA), self-organizing map neural networks (SOM), and Simulated Annealing (SA) are employed to find near-optimal solutions more efficiently.

To explore and implement effective solutions, the research addresses the following questions: 1) How can route optimization be achieved for the replacement of kWh meters in the context of the Traveling Salesman Problem (TSP)? 2) How do the methods of Dynamic Programming, Genetic Algorithm, and Google OR Tools compare in optimizing the replacement routes for kWh meters?

In addressing the optimization of kWh meter replacement routes, this research will primarily employ Genetic Algorithms (GA) due to their flexibility and effectiveness in solving complex optimization problems. Genetic Algorithms mimic the process of natural evolution, utilizing selection, crossover, and mutation to iteratively improve solutions. This method is particularly suitable for handling large-scale and complex routing problems, as it can efficiently explore a vast search space to find near-optimal solutions.

To ensure a comprehensive evaluation and robustness of the optimization process, this research will also incorporate Google OR-Tools and Dynamic Programming as comparative and alternative methods. Google OR-Tools, a versatile and powerful optimization toolkit, offers various algorithms specifically designed for solving routing problems and provides a practical implementation framework. Dynamic Programming, known for its ability to break down problems into simpler subproblems, will be used to find exact solutions for smaller instances and serve as a benchmark for evaluating the performance of other methods. By comparing these techniques, the study aims to identify the most effective approach for optimizing kWh meter replacement routes under the constraints faced by ULP Tegalrejo.

The goal of this independent study is to find the optimal method for scheduling timely APP replacement routes using Dynamic Programming, Genetic Algorithms (GA), and Google OR-Tools. The benefits of this research include helping PLN address the challenges in enhancing resource management efficiency in the kWh meter replacement process.

II. RELATED WORK

Previous research by [1] attempted to solve the TSP in India using Genetic Algorithms (GA), self-organizing map neural networks (SOM), and Simulated Annealing (SA). They found GA to be a reliable and efficient algorithm for finding optimized routes. One reason GA is considered attractive is its support for global optima and ease of implementation [2]

Given the complexity and scale of the problem at ULP Tegalrejo, Genetic Algorithms will be used to optimize the replacement routes, ensuring efficient execution and near-optimal solutions. Additionally, Google OR-Tools and Dynamic Programming will be utilized for comparison, offering alternative methods to evaluate and ensure the robustness of the route optimization process. These tools will help address the challenges associated with large-scale route optimization in the APP replacement program, ultimately improving efficiency and meeting the ambitious replacement targets.

Research by [3] compared the performance of a genetic algorithm in solving the Traveling Salesman Problem (TSP) by implementing two different types of crossover: Static Crossover and Dynamic Crossover. The flowchart depicted in Figure 1 outlines the process. After selecting two parents, the crossover process is conducted twice for comparison purposes. The first crossover, termed Static Crossover, applies two fixed crossover points in each iteration. In contrast, the second crossover, termed Dynamic Crossover, selects two random positions in each iteration. The crossover process at points 1 and 4 is shown in Figure 2. Statistical analysis of the data generated by the proposed algorithm revealed that Dynamic Crossover was faster than Static Crossover in finding solutions. Moreover, Dynamic Crossover significantly outperformed Static Crossover in identifying the shortest path. The performance results are summarized in Figure 3. This study highlights the efficiency of Dynamic Crossover in enhancing the speed and quality of solutions obtained by genetic algorithms for solving the TSP.

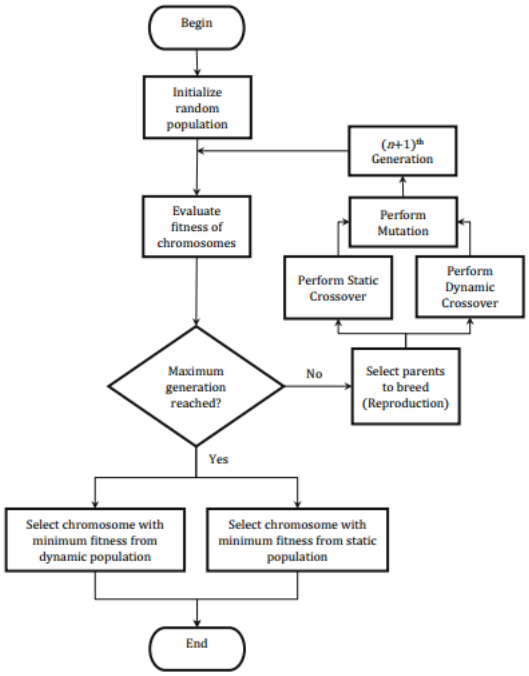

Fig. 1. Genetic Algorithm Flowchart Research by [3]

III. PROBLEM FORMULATION

The problem at hand involves optimizing the kWh meter replacement routes can be modeled as a Traveling Salesman Problem (TSP). Defining each route as graph $R = (V, A)$ where V repesents the set of vertices i and A represents the set of arcs (i, k)

- A set of N location, $L = \{l_1, l_2, l_3, \dots, l_N\}$ where N=52,071 repesenting the customer locations.
- A distance matrix $D = [d_{ij}]$ where $[d_{ij}]$ denotes the distance between location l_i and l_j.
- The starting and ending point of the route is the same location (the depot).

The objective is to find the shortest possible route that allows a technician to visit each customer exactly once and return to the starting point distance as seen in the objective function defined in (1).

$$Minimize = \sum_{i=1}^{N-1} d_{x_i x_{i+1}} + d_{x_N x_1} \quad (1)$$

Subject to:

$$\sum_{j=1, j\neq i}^{N} x_{ij} = 1 \qquad \forall_i = \{1,2,3, \dots, N\} \quad (2)$$

$$\sum_{i=1, j\neq i}^{N} x_{ij} = 1 \qquad \forall_i = \{1,2,3, \dots, N\} \quad (3)$$

$$x_{start} = x_{end} = \text{PLN ULP Tegalrejo Office} \quad (4)$$

Where x_i is the i-th location in the sequence of visits. This formulation ensures the total travel distance is minimized, thereby optimizing the kWh meter replacement routes. Additionally, formula (2) ensures that each customer is visited only once, and formula (3) ensures that the starting and ending point of the route is the PLN office.

IV. PROPOSES APPROACH

In order to mininized total travel disatnce of the kWh meter replacement routes, we study three methods that we referred as: Dyna mic Programming ,Genetic Algorithms (GA), and Google OR-Tools.

A. *Dynamic Programming*

The architecture for solving the TSP using Dynamic Programming involves dividing the problem into smaller subproblems by considering all possible combinations of cities to visit and storing the results to avoid redundancy. It then calculates the minimum distance required to reach each city in every combination iteratively, taking into account the cities that have already been visited. Finally, it selects the shortest route that visits all cities exactly once and returns to the starting point.

B. *Genetic Algorithm*

In general, the solution design flow created is similar to the explanation by [4]. The architecture for solving the TSP using Genetic Algorithm (GA) illustrates in Fig. 2.

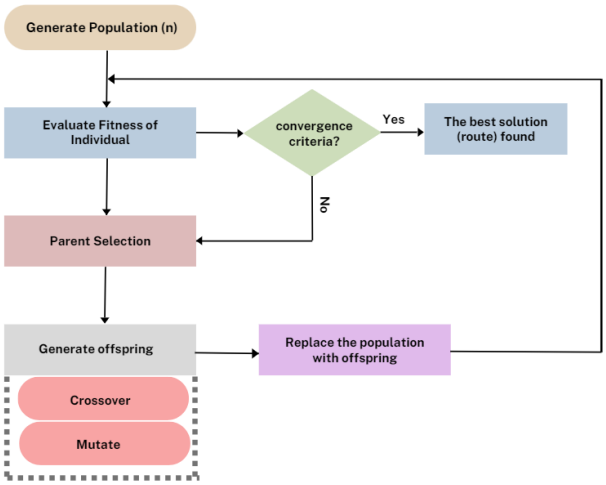

Fig. 2. Genetic Algorithm Flowchart

The population consists of a set of individuals, each representing a solution to the problem at hand. Each solution starts from node 1, representing the common departure point: PLN ULP Tegalrejo. The individuals are stored in a list containing two main elements: the fitness value and the detailed travel route.

There are two approaches for solving the TSP using the Genetic Algorithm differ in several key aspects. Both approaches start by initializing the population with a random order of cities (random route is generated), except for the first city, which remains fixed. For selection, the first approach uses roulette wheel selection based on fitness, whereas the second approach employs tournament selection.

Crossover operations also differ between the two approaches. The first approach utilizes two-point random crossover, while the second approach uses a single-point random crossover with a probability of 0.9. If crossover does not occur in the second approach, the offspring will entirely copy one parent. For mutation, both approaches swap two cities randomly, but with differing mutation probabilities 0.7 for the first approach and 0.1 for the second approach.

Elitism, which ensures the best solutions are carried over to the next generation, is not used in the first approach but is implemented in the second approach. Despite these differences, both approaches maintain the same population size of 2,000, a crossover rate of 0.9, and differ only in their mutation rates. The second approach is based on the code by [5] while the first approach is developed independently following the methodology described by [4]. These variations are designed to compare the effectiveness and efficiency of the different genetic algorithm configurations in optimizing the TSP route.

C. *Google OR Tools*

Google OR-Tools is an open-source software suite developed by Google for optimization tasks, including solving the Traveling Salesman Problem (TSP). The process of solving TSP with Google OR-Tools involves several technical steps. First, a distance matrix is created by calculating the Euclidean distance between each pair of points (cities), which forms the basis of the distance matrix. Next, this distance matrix is inputted into Google OR-Tools, which provides an interface to define the number of vehicles (usually one for TSP) and the starting point (depot).

V. EXPERIMENTS AND RESULTS

A. *Dataset*

The dataset used in this study comprises customer data from PLN ULP Tegalrejo, where kWh meters are scheduled for replacement. The data includes Customer ID, Customer Name, Longitude, Latitude, and Address. To simplify the problem, Customer ID will be replaced with a Sequence Number. Customer Name and Address will not be used in this analysis. The data for Sequence Number, Longitude, and Latitude will be stored in a list with the following format:

$$[[no\_customer1, Latitude1, Longitude1],$$

$$[no\_customer2, Latitude2, Longitude2],$$

$$\ldots,$$

$$[no\_urut\_pelanggan\_n, Latitude, Longitude]]$$

To calculate the distance, the Cartesian distance between each city and the next in a circular route is used. The Cartesian distance between city A and city B is calculated using the Euclidean distance formula as follows:

$$A - B = \sqrt{(Lat\_A - Lat\_B)^2 + (Lot\_A - Lot\_B)^2}$$

B. *Experimental Result*

To compare the effectiveness and accuracy of the implementations of Dynamic Programming, Genetic Algorithm (with two approaches), and Google OR-Tools in solving the Traveling Salesman Problem (TSP), several tests were conducted. Initially, tests were performed on TSP cases with a small number of cities to assess the ability of each algorithm to find optimal solutions in relatively simple scenarios. Subsequently, tests were carried out on TSP cases with a larger number of cities to evaluate the performance of the algorithms in handling more complex problem scales.

The results are presented in the Table 1 which shows the comparison of optimal distances, while Table 2 provides the execution times for each algorithm with different numbers of points.

Additionally, execution time efficiency tests were conducted for each algorithm using various test cases with differing sizes and complexities. This aimed to assess how quickly each algorithm could solve the TSP under practical conditions. Lastly, tests were performed to compare the limitations of each algorithm, such as the maximum number of cities they could handle, to understand the constraints present in their usage.

TABLE I. COMPARISON OF OPTIMAL DISTANCES

Number of Points	Distance (Euclidiance Distance)			
	DP	GA 1	GA 2	*Google OR Tools*
5	0.2615118544	0.2615118544	0.2615118544	0.217
10	0.2167872	0.2910063971	0.2952655081	0.216
20	0.3253422	0.6091712839	0.6103369923	0.324
25	crash	1.482172154	1.439733731	0.969
50		3.156612042	3.449460015	1.173
100		7.546015092	7.301396955	1.458
250		18.59628198	18.44997906	1.721
500		38.30809253	38.50418933	2.069

TABLE II. COMPARISON OF EXECUTION TIME

Number of Points	Execution Time (*Seconds*)			
	DP	GA 1	GA 2	*Google OR Tools*
5	0.00018143654	1.147232771	0.0674161911	0.02074360847
10	0.02988648415	1.471156359	0.0711183548	0.0042
20	66.68308711	1.788134336	0.05134248734	0.0186
25	crash	6.220061302	0.07561922073	0.0099
50		454.1269572	0.1296982765	0.3033
100		1626.022489	0.3593554497	1.0754
250		2005.296444	2.150679111	4.7053
500		14685.08040	6.872550488	16.7794

C. Evaluation and Analysis

Dynamic Programming (DP) is a method that ensures an optimal solution by breaking down the problem into smaller subproblems and storing the results for reuse. This method is highly effective for problems with a small number of points, such as 5 or 10 points, where DP provides very good results with consistent optimal distances. However, the main drawback of DP is its scalability. As the number of points increases, the computational complexity and memory requirements grow exponentially. This leads to very long execution times and even crashes when the number of points reaches 25 or more.

The significant differences in the test results between the two Genetic Algorithm (GA) approaches are due to differences in their methods of selection, crossover, mutation, and elitism. The first approach uses roulette wheel selection based on fitness, two-point random crossover, and does not implement elitism. The roulette wheel selection method allows solutions with higher fitness to have a greater chance of being selected as parents, but it can cause premature convergence if high-fitness solutions dominate too early. Two-point random crossover can produce offspring with a combination of genes from both parents but often introduces many changes that may not always be beneficial. Additionally, without elitism, the best solutions can be lost in the next generation, leading to a decline in overall solution quality.

In contrast, the second approach uses tournament selection, single-point random crossover with a probability of 0.9, mutation with a lower probability of 0.1, and implements elitism. Tournament selection provides better control over selection pressure and helps maintain population diversity for a longer period. Single-point random crossover, though simpler, produces offspring with more stable gene combinations, reducing the risk of unfavorable drastic changes. Mutation with a lower probability helps maintain gene stability within the population while still providing enough variation for solution space exploration. The use of elitism ensures that the best solutions are retained for the next generation, improving the overall solution quality.

The test results show that the second approach is superior in terms of optimal distance and execution time. This approach yields better results than the first approach in finding more efficient and effective solutions while maintaining solution quality with shorter distances. The second approach also demonstrates lower execution times, especially for a larger number of cities, as tournament selection, single-point random crossover, and elitism offer better computational efficiency. Thus, the second approach is more effective in finding Pareto-optimal solutions efficiently and maintaining population diversity to avoid premature convergence.

Google OR-Tools consistently excels in terms of optimal distance and execution time. This outstanding performance makes it the most efficient and effective choice for route optimization in this test. The flexibility, scalability, and computational efficiency of Google OR-Tools make it highly suitable for route optimization problems with a large number of points, providing solutions that are not only accurate but also fast in execution time.

VI. CONCLUSIONS AND FUTURE WORK

Based on the test results, it can be concluded that Google OR-Tools demonstrated excellent performance in optimizing the kWh meter replacement routes at ULP Tegalrejo, both in terms of optimal distance and execution time. This algorithm consistently provided the shortest distance solutions and the fastest execution times for various numbers of points, making it a highly efficient and effective choice.

The second approach of the Genetic Algorithm (GA), which utilizes tournament selection, single-point random crossover with a probability of 0.9, and elitism, showed better results compared to the first approach, which used roulette wheel selection, two-point random crossover without a specific probability, and no elitism. The second approach delivered solutions with shorter distances and faster execution times, especially for a larger number of points.

Meanwhile, the Dynamic Programming method proved effective for small cases but was not suitable for cases with a large number of points due to high computational complexity and significant memory requirements.

Further research can be conducted to explore other optimization methods such as Simulated Annealing or Particle Swarm Optimization to compare results and find the most efficient and effective method for various scenarios. Additionally, using real-world distances obtained from Google Maps in the optimization process is suggested to enhance the accuracy and applicability of the solutions.

Future work could also focus on dividing the main route into sub-routes that account for the working hours of the staff and optimizing these sub-routes to ensure an even distribution of workload in accordance with the available working hours.

REFERENCES

[1] A. R. Khaparde, S. K. Pillai, A. Saini, U. Agrawal, B. Balamurugan, and G. Dhuriya, “National Travelling Salesman Problem for Indian Cities: a Case Study,” in *Proceedings of the 2022 3rd International Conference on Intelligent Computing, Instrumentation and Control Technologies: Computational Intelligence for Smart Systems, ICICICT 2022*, Institute of Electrical and Electronics Engineers Inc., 2022, pp. 717–720. doi: 10.1109/ICICICT54557.2022.9917908.

[2] U. Mehboob, J. Qadir, S. Ali, and A. Vasilakos, “Genetic Algorithms in Wireless Networking:

Techniques, Applications, and Issues," Nov. 2014, [Online]. Available: http://arxiv.org/abs/1411.5323

[3] T. Alzyadat, M. Yamin, and G. Chetty, "Genetic algorithms for the travelling salesman problem: a crossover comparison," *International Journal of Information Technology (Singapore)*, vol. 12, no. 1, pp. 209–213, Mar. 2020, doi: 10.1007/s41870-019-00377-9.

[4] P. Norvig and S. Russell, "Artificial Intelligence A Modern Approach Third Edition," 2010.

[5] M. Hassanzadeh, "Traveling-Salesman-Problem-using-Genetic-Algorithm." Accessed: Jun. 01, 2024. [Online]. Available: https://github.com/hassanzadehmahdi/Traveling-Salesman-Problem-using-Genetic-Algorithm

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 12 Juni 2024

IKE AYU IDIARA
23522306