Evaluating Distribution Strategy of Makan Bergizi Gratis at SPPG Cimanggung Using The Ant Colony Optimization

Cherinette Corsane Khassyah Purceria - 13525003
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail: cherinettekhassyah@gmail.com , 13525003@std.stei.itb.ac.id

***Abstract*— Makan Bergizi Gratis (MBG) is a national free nutritious meal program established under the administration of Prabowo Subianto's, the 8th Indonesian President. This program is managed locally by a Nutritional Provision Service Unit (Satuan Pelayanan Pemenuhan Gizi or SPPG) of each area and targets school pupils and pregnant woman. However, this program has experienced cases of poisoning food. This is caused by ineffective distribution time, making food freshness as a critical challenge for this program. This paper will examine the distribution strategy of one SPPG in Cimanggung, Sumedang, West Java using Ant Colony Optimization to prevent future risks. The distribution network is modeled as a complete weighted graph, where edge weights represent travel time. The ACO algorithm then implemented to the graph to determine most time-efficient route. ACO-optimized route shows a 7.1% improvement over the current route.**

Keywords—Makan Bergizi Gratis, Ant Colony Optimization, Distribution Strategy.

I. INTRODUCTION

Food poisoning incidents in Indonesia's national program Makan Bergizi Gratis (MBG), or Free Nutritious Meal, have become a serious public concern. Recorded by Jaringan Pemantau Pendidikan Indonesia (JPPI), this program has resulted in 33,626 reported poisoning cases since its launch in 2025, with the number increasing each month and the location of cases spread all across Indonesia, with West Java recording the highest number of incidents compared to other provinces.

These cases of poisoning stem from various factors, including bacterial contamination, nitrite exposure, and others. All of those factors arise from weak safety standards of operational. Every step of MBG operational chain has the risk of contributing to contamination, including the distribution stage. According to BGN's Standard Operating Procedure, meal must not exceed 30 minutes to be delivered. This requirement highlights the critical need for an effective distribution strategy.

In order to achieve an effective distribution time, SPPG must take the shortest path to minimize distribution time. Route determination can be modeled using graph theory, where distribution points are represented as vertices and travel times between locations as weighted edges. Among various optimization algorithms, Ant Colony Optimization (ACO) is selected due to its proven effectiveness in solving combinatorial routing problems, particularly the Vehicle Routing Problem (VRP). This problem further formulated as a Vehicle Routing Problem with Time Windows (VRPTW), since every recipient must receive the meals within a specific time frame.

This paper aims to determine the most time-efficient delivery route for SPPG Cimanggung in distributing MBG to recipients. Thus, the results can be served as a basis for evaluating the current distribution route and preventing future risks.

II. THEORITICAL BACKGROUND

A. Graph Theory

A.1. Graph Theory

Graph is a structure consisting of vertices that connected by edges, used for represents connected objects. A graph G is defined as G = (V, E), where V is a non-empty set of vertices $\{v_1, v_2, \ldots, v_n\}$ and E is set of edges $\{e_1, e_2, \ldots, e_n\}$. Relation of edges and vertices denoted as e = (u, v), where e is an edge that connects vertices u and vertices v.

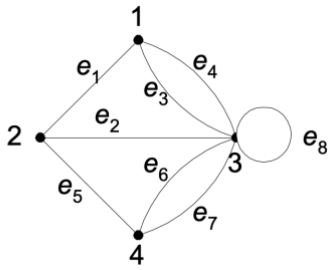

Fig 2.1 Graph
(Source: [6])

Based on the edge's direction, graphs are classified into two types, undirected graph and directed graph. An undirected graph is a graph with the edges have no orientation, where the edge (u, v) is identical to (v, u). While directed graph has a specific orientation of each edge, denoted as edge (u, v) indicating a one-way path from vertex u to vertex v.

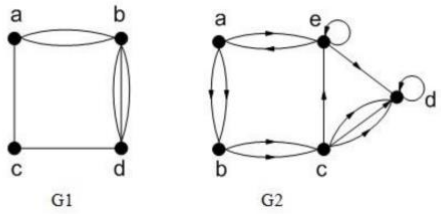

Fig 2.2 Undirected Graph and Directed Graph
(Source: [6])

Based on the edge's weight, graphs are also classified into two types, unweighted graph and weighted graph. An unweighted graph is a graph with all edges are considered equal, where all edges indicates a relationship without any value assigned to the path. While weighted graph is a graph with each edge is assigned a value. This value can be used to represent cost, distance, time, and others.

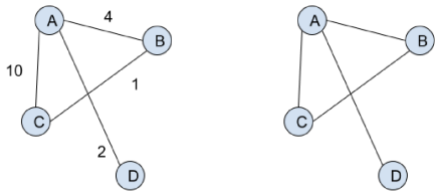

Fig 2.3 Weighted Graph and Unweighted Graph
(Source: [6])

A.2. Graph Terminologies

To model a connecting objects, several terminologies are written to represent conditions in a graph. Here are some terminologies that used in this research.

Adjacent, two vertices that are connected are said to be adjacent. In the figure below, vertex 1 has adjacent with vertex 2 and vertex 3. However, vertex 1 has not adjacent with vertex 4.

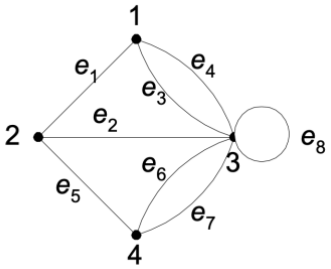

Fig 2.4 Graph with Incidents
(Source: [6])

Incidency, for any given edge e = (u, v), the edge e is said to be incident with vertex e and vertex v. Based on figure above, edge e1 is incident with vertex 1 and vertex 2, while edge e2 is incident with vertex 2 and vertex 3.

Degree, represents number of edges that are incident with a given vertex, notated as d(v). For example, in the figure below, d(1) = 2 and d(5) = 0.

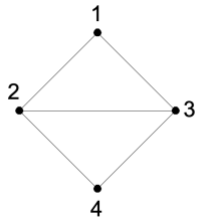

Fig 2.1 Graph with Adjacent Vertices
(Source: [6])

In directed graph, degrees are classified into in-degree and out-degree. Based on figure below, $d_{in}(1) = 2$ and $d_{out}(1) = 1$.

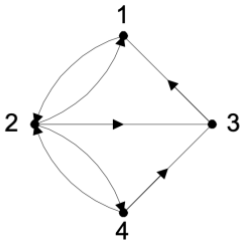

Fig 2.5 Directed Graph
(Source: [6])

Path, a path of length n from an initial vertex v_0 to vertex v_n in a graph G is an alternating sequences of vertices and edges expressed as $v_0, e_1, v_1, \ldots, v_{n-1}, e_n, v_n$, where each consecutive edge satisfies $e_k = (v_{k-1}, v_k)$ for k = 1, 2, …, n.

Cycle, a path that originates and terminates at the same vertex. Cycle is also known as circuit.

Connected, a graph is said to be a connected graph if every pair of vertex vi and vj inside set V has a path from vi to vj. If a graph does not have a path from vi to vj, then it said to be an unconnected graph.

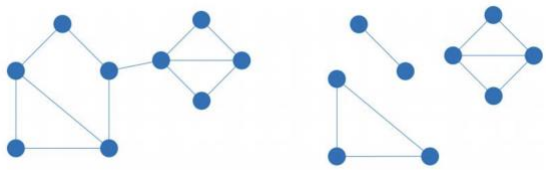

Fig 2.6 Connected Graph and Unconnected Graph
(Source: [6])

Subgraph, a graph $H = (V_H, E_H)$ is defined as a subgraph of $G = (V_G, E_G)$ if VH is a subset of V_G and E_H is a subset of EG.

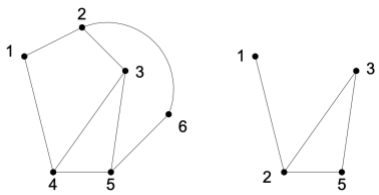

Fig 2.6 Graph and Its Subgraph
(Source: [6])

B. *Ant Colony Optimization*

Ant Colony Optimization (ACO) is a technique to approximate optimization inspired by real ant colonies. ACO represents ants' indirect communication by means of chemical pheromone trails, which enables them to determine shortest paths between their nest and food sources. When searching food, ants initially explore the area surrounding their nest. Once a food is located, ant evaluates the quality and quantity of the food, which then carried to its nest. During the return journey, the amount of pheromone laid down on the surface varies based on the food's assessed value. Consequently, these established chemical trails serve as a navigational guide for other colony members toward the food supply. The first version of algorithm was introduced by Marco Dorigo in the early 1990's.

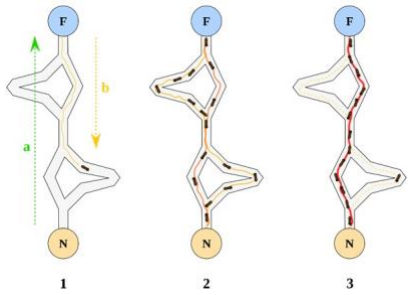

Fig 2.7 Ant Colony Optimization Ilustration
(Source: [5])

This ant behavior modelled as a graph G = (V, E), where V represents the nest and all possible food sources, while E represents the paths between. To guide ant's decision, a value represents pheromone is assigned to each edge, making graph G into a weighted graph. The weighted graph then serves as a foundation of the entire ACO process.

Pheromone trail denoted as τ_{ij}. All edges are initialized with an equal pheromone value, $\tau_{ij}(0) = \tau_0$. Each edge is also assigned with a value, ηij, represents the desirability of traversing that edge regardless of the pheromone history, commonly defined as the inverse of the edge weight, $\eta_{ij}=\frac{1}{\tau_{ij}}$ Therefore, edges with lower cost are considered more attractive. In constructing a path, an ant positioned at vertex I selects the next vertex j from the set of unvisitied vertices based on the probability function,

$$P_{ij}^{k} = \frac{[\tau_{il}]^{\alpha} \cdot [\eta_{il}]^{\beta}}{\sum_{l \epsilon \mathcal{N}_i^k} [\tau_{il}]^{\alpha} \cdot [\eta_{il}]^{\beta}}$$

Where α and β are the parameters controlling the influence of pheromone and desirability. A higher value of α causes ant to rely more on accumulated pheromone trails, while a higher value β causes ant to selects edges with lower weight regardless of pheromone history.

After all iterations are completed, the pheromone levels on the graph are updated through two sequential processes, evaporation and deposition. Evaporation means reducing the pheromone level on every edges by a constant factor, while deposition means the amount of pheromone on every edge traversed during ant's journey, proportional to the quality of the solution it constructed. Evaporation is expressed as,

$$\tau_{ij} \longleftarrow (1-\rho) \cdot \tau_{ij}$$

Where ρ is the evaporation rate, valued between 0 and 1. Deposition is expressed as,

$$\tau_{ij} \longleftarrow \tau_{ij} + \sum_{k=1}^{m} \Delta\tau_{ij}^{k}$$

$$\Delta\tau_{ij}^{k} = Q/T^{k}$$

Where m is the number of ants used in each iteration, Q is a constant that scales the amount of pheromone deposited and T^k is the total cost of the tour constructed by ant k. These steps will be iterated T_{max} times and stops with its best route.

C. *Vehicle Routing Problem*

The vehicle routing problem (VRP) is a combinatorial optimization problem concerned with determining a set of routes for a fleet of vehicles, originating and terminating at a central depot, with the minimum distance or cost. The VRP is one of the problem domains to which ACO is applied, since both share the same underlying graph-based structure. Bell and McMullen (2004) demonstrated this relationship by adapting the ant colony procedure originally developed for the Traveling Salesman Problem to solve the Vehicle Routing Problem, modifying the construction process so that an ant returns to the depot once the vehicle's capacity constraint is met before initiating a new route. A further research of the VRP results The Vehicle Routing Problem with Time Windows (VRPTW). In this case, delivery to each customer must occur within a specific time window. The VRP is solved under these 3 conditions,

- Each customer is visited only once by a single vehicle
- Each vehicle must start and end its route at the depot
- Total demand serviced by each vehicle does not exceed Q, where Q is a capacity constraint.

III. PROPOSED METHOD

A. *Route Planner*

SPPG Cimanggung delivers meal every morning to 11 recipients using 2 vehicles. This paper focuses on the first vehicle which delivers 8 of 11 recipients, taking around 85 minutes. The distribution network of SPPG Cimanggung represented as a graph, with each location modeled as the vertices and edges connecting those vertices represents the road between two locations. Moreover, each edge is assigned a value estimated travel time required to traverse it. Location of the SPPG and recipients were obtained through an interview from the SPPG directly. Travel time of 8 locations were obtained from Google Maps by adjusting the road traffic at 7 AM.

Tabel 1. Recipient List

Vertex	Location
0	SPPG Cimanggung
1	SDN Cimanggung IV
2	TK Cikalama II
3	KB Aisyiyah
4	TK Miftahul Jannah
5	TK Dangiang Danarasa
6	TK Zad Al-Khoir
7	SLB Al-Faza
8	Posyandu Mawar

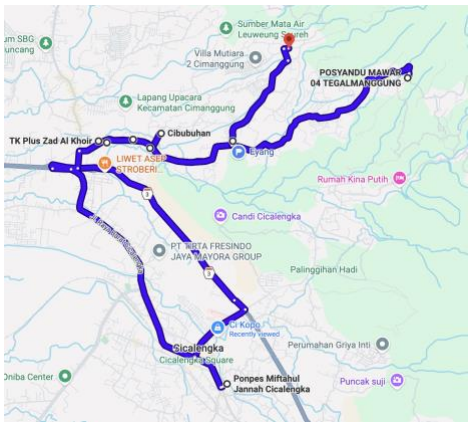

Fig 3.1 Locations of Recipients
(Source: [3])

This study uses travel time, instead of distance travelled, as the edge weight for the graph model since the central concern of this study is food safety, such as the BGN SOP's maximum delivery time and the Ministry of Health's four-hour food safety limit, rather than distance-based standards. Below is the adjacency matrix of the graph, represents the weight of each edge in minutes.

Table 2. Adjacency Matrix

V	0	1	2	3	4	5	6	7	8
0	0	1	3	14	15	4	15	15	9
1	1	0	4	15	15	4	15	15	9
2	3	4	0	11	11	1	12	11	12
3	4	15	11	0	23	12	13	12	13
4	15	15	11	23	0	11	20	20	24
5	4	4	1	12	11	0	11	11	13
6	5	15	12	13	20	11	0	11	14
7	5	15	11	12	20	11	11	0	14
8	9	9	12	13	24	13	14	14	0

The graph constructed in this study is a complete graph, which every pair of distinct vertices is connected by exactly one edge. This property used due to the delivery vehicle is free to travel from any location to any other location in any order, a direct route between every pair of nodes must exist and be assigned a defined travel time, regardless of whether that particular connection is ultimately used in the optimized route. Figure below illustrates the graph constructed from Table 2.

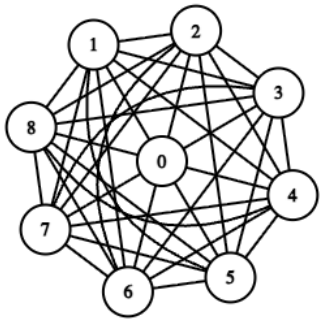

Fig 3.2 Model of SPPG Cimanggung Distribution Network
(Source: csacademy.com)

B. ACO Implementation

After weighted graph is constructed, parameter of ACO algorithm has to be determined before implementing ACO algorithm. Based on the probability formula and mechanism of evaporation and deposition, there are seven parameters that need to be determined, $\tau_0, \alpha, \beta, \rho, m, Q,$ *and* T_{max}. Based on previous study on VRP, good parameter values for the influence of pheromone and heuristic desirability are $\alpha = 1.0$ and $\beta = 2.0$. Furthermore, the initial pheromone will be set to τ_0=1.0 and the evaporation rate to $\rho = 0.5$, while the number of ant used in each iteration is set to m = 20, with parameter Q is set to 100, and the algorithm runs for maximum of T_{max} = 100.

ACO algorithm begin with initialization of pheromone and heuristics value, followed by construction by each ant using probability formula based on heuristic and pheromone intensity. Pheromones updated through evaporation and deposition, allowing the algorithm to converge toward the optimal route after reaching the maximum number of iterations.

```
tau = []
for i in range(n):
    tau_row = []
    for j in range(n):
        if i == j:
            tau_row.append(0)
        else:
            tau_row.append(TAU0)
    tau.append(tau_row)

eta = []
for i in range(n):
    eta_row = []
    for j in range(n):
        if i == j:
            eta_row.append(0)
        else:
            eta_row.append(1 / W[i][j])
    eta.append(eta_row)
```

Fig 3.3 Calculating All τ and η
(Source: Author's own computation)

The first block of code above builds a $n \times n$ matrix of pheromone and setting every edge to τ_0, except for the diagonal matrix. The second block also builds $n \times n$ matrix of heuristic desirability by using the invers of the edge's weight.

```
for iteration in range(T_MAX):
    all_routes = []
    all_times = []

    for k in range(M):
        route = [0]
        visited = [0]
```

Fig 3.4 Main Iteration of Program
(Source: Author's own computation)

This iteration is the core structure of the ACO algorithm. The outer loop repeats the entire route construction process for Tmax times, with the inner loop runs m ants constructing one complete candidate route per iteration. The variable all_routes and all_times are reset to empty at the start of every iteration to stores only the routes and travel times made by ants in one iteration. Only pheromone matrix carries information forward for the next iteration.

```
scores = []
for j in candidates:
    tau_value = tau[current][j] ** ALPHA
    eta_value = eta[current][j] ** BETA
    scores.append(tau_value * eta_value)
```

Fig 3.5 Calcualting Scores
(Source: Author's own computation)

This block calculates how desirable each possible next location is for the ant that currently standing at vertex current. It does the calculation for every location in candidates, which includes all locations the ant has not visited yet.

```
probabilities = []
for s in scores:
    if total_score == 0:
        probabilities.append(1 / len(candidates))
    else:
        probabilities.append(s / total_score)
```

Fig 3.6 Calculating Probabilities
(Source: Author's own computation)

This block takes scores calculated in previous step and converts them into probabilities by divides each score by total_score, which translates to the probability formula explained in the previous section. By the end of this loop, probabilities holds one value for each candidate location to be used in the next step where ant randomly picks its next destination.

```
random_number = random.random()
cumulative = 0
chosen = candidates[len(candidates) - 1]

for idx in range(len(candidates)):
    cumulative = cumulative + probabilities[idx]
    if random_number <= cumulative:
        chosen = candidates[idx]
        break
```

Fig 3.7
(Source: Author's own computation)

The code above shows how the ant picks the next node, weighted by probability calculated before. A random number between 0 and 1 is generated and the code adds up the probabilities one by one until the running total passes the random number. Locations with a higher probability will more likely to be picked.

After every ant has finished building its route using the probability values described above, the algorithm moves on to evaporation and deposition. This step happens after all m ants have completed their routes for the current iteration.

```
for i in range (n):
    for j in range (n):
        if i != j:
            tau[i][j] = (1 - RHO) * tau[i][j]
```

Fig 3.8 Calculating Evaporation
(Source: Author's own computation)

```
for idx in range(len(ant_route) - 1):
    origin = ant_route[idx]
    destination = ant_route[idx + 1]
    tau[origin][destination] = tau[origin][destination] + (Q / ant_time)
```

Fig 3.9 Calculating Deposition
(Source: Author's own computation)

Evaporation reduces the pheromone on every edge. Deposition comes right after, adding more pheromone to the edges with shorter routes. These two steps let the pheromone matrix shift over many iterations then produce shorter routes, which allows the algorithm to improve over time.

IV. RESULT

The optimized route found by the ACO algorithm is V0→V1→V2→V5→V4→V7→V6→V8→V3→V0, with total travel time of 79 minutes. The current route of meal distribution at SPPG Cimanggung is V0 →V1→V2→V3→V4→V5→V6→V7→V8→V0, which takes 85 minutes. Compared to the route determined by ACO algorithm, current route is 6 minutes slower. This result represent a 7.1% improvement over the current route practiced by SPPG Cimanggung.

```
ACO RESULT - TSP MBG Distribution SPPG Cimanggung

Best route (vertex numbers):
 0 -> 1 -> 2 -> 5 -> 4 -> 7 -> 6 -> 8 -> 3 -> 0 ->
Best route (location names):

SPPG Cimanggung -> SDN Cimanggung IV -> TK Cikalama II -> TK Dangiang Danarasa
 -> TK Miftahul Jannah -> SLB Al-Faza -> TK Zad Al-Khoir -> Posyandu Mawar ->
KB Aisyiyah -> SPPG Cimanggung ->


Optimal total travel time: 79 minutes

Breakdown of each segment:
SPPG Cimanggung -> SDN Cimanggung IV : 1 minutes
SDN Cimanggung IV -> TK Cikalama II : 4 minutes
TK Cikalama II -> TK Dangiang Danarasa : 1 minutes
TK Dangiang Danarasa -> TK Miftahul Jannah : 11 minutes
TK Miftahul Jannah -> SLB Al-Faza : 20 minutes
SLB Al-Faza -> TK Zad Al-Khoir : 11 minutes
TK Zad Al-Khoir -> Posyandu Mawar : 14 minutes
Posyandu Mawar -> KB Aisyiyah : 13 minutes
KB Aisyiyah -> SPPG Cimanggung : 4 minutes
```

Fig 4.1 Program Output
(Source: Author's own computation)

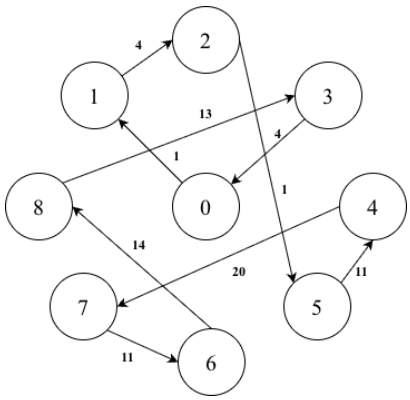

Figure 4.2 Directed Graph Resulted by ACO Algorithm (Source:

To evaluate the stability and efficiency of the ACO program, a convergence checkpoints were outputted over iterations. The rate of convergence demonstrates how rapidly the virtual ants discover the global minimum path.

```
Convergence checkpoints:
Iteration 1 -> best_time: 79
Iteration 3 -> best_time: 79
Iteration 10 -> best_time: 79
Iteration 25 -> best_time: 79
Iteration 50 -> best_time: 79
Iteration 100 -> best_time: 79
```

Fig 4.3 ACO Convergence Checkpoints

Unlike a typical convergence curve that gradually improves over iterations, the output shows that virtual ants was already found the shortest path in the first iteration. Since there are only 9 vertices to connect, the number of possible routes is small. Because of this, one of the 20 ants could find the shortest path in the very first step by quick calculation, without needing pheromone trails to guide the group yet. This shows the code is working well for small data, but less representative of how ACO performs on large and more complex problems.

V. CONCLUSION

This paper evaluated the distribution strategy of Makan Bergizi Gratis at SPPG Cimanggung by implementing Ant Colony Optimization algorithm. The experimental results demonstrate that the proposed method successfully optimizes the total travel time compared to the current conventional routing. This finding confirms that ACO algorithm can effectively minimize logistics latency in school meal distribution networks. However, in real life, the optimized delivery time is still too long to meet the safety standard set by the National Nutrition Agency (BGN). Therefore, a revision of the system must be re-examined. For example, splitting the delivery among multiple vehicles.

APPENDIX

The python program of ACO can be accessed through this link, https://github.com/ccpurceria/Ant-Colony-Optimization-for-SPPG-Cimanggung.git

ACKNOWLEDGMENT

The author would like to express her gratitude to Almighty God, whose grace and guidance turned an overwhelming academic challenge into a journey of discovery. The author would also like to thank the lecturer of IF1220 Descrete Mathematic, Mr.Rinaldi Munir, for all the knowledge taught throughout the semester. Lastly, the author extends her appreciation to her family and friends who always provide endless support and prayers during this process. However, the author fully realizes that this paper is still far from perfection and leaves room for further development. Therefore, criticisms, insights, and suggestions are openly welcomed for its future improvement.

REFERENCES

[1] C. Blum, "Ant colony optimization: Introduction and recent trends," Physics of Life Reviews, vol. 2, no. 4, pp. 353–373, Dec. 2005.

[2] J. E. Bell and P. R. McMullen, "Ant colony optimization techniques for the vehicle routing problem," Advanced Engineering Informatics, vol. 18, no. 1, pp. 41–48, Jan. 2004.

[3] Google, "Google Maps Satellite View of Cimanggung Area", Google Maps, Jun. 2026. [Online]. Available: https://www.google.com/maps [Accessed: Jun. 18, 2026].

[4] H. A. H. Hasan and A. S. Ahmed, "Development of a new approach algorithm for ant colony optimization using Python program," Journal of Interdisciplinary Mathematics, vol. 28, no. 1, pp. 95–105, 2025, doi: 10.47974/JIM-1800.

[5] M. D. Toksari, "A hybrid algorithm of Ant Colony Optimization (ACO) and Iterated Local Search (ILS) for estimating electricity domestic consumption: Case of Turkey," *International Journal of Electrical Power & Energy Systems*, vol. 78, pp. 276–282, June 2016, doi: 10.1016/j.ijepes.2015.12.032)

[6] Munir, Rinaldi. 2024. "Graf Bagian 1". https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/20-Graf-Bagian1-2024.pdf. [Accessed on June 17, 2026].

[7] [15] H. Hanifah, "33 Ribu Pelajar Keracunan MBG, Target 3000 Porsi per SPPG Dinilai Melebihi Kapasitas," *Universitas Gadjah Mada*, Apr. 24, 2026. [Online]. Available: https://ugm.ac.id/id/berita/33-ribu-pelajar-keracunan-mbg-target-3000-porsi-per-sppg-dinilai-melebihi-kapasitas/ [Accessed: Jun. 19, 2026].

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.

Sumedang, 19 Juni 2026

Cherinette Corsane Khassyah Purceria - 13525003