Vol.6 No.3 July 2023

Multimedia Research

Received 08 May, Revised 27 May, Accepted 27 June

Optimization of the Ant Colony System Algorithm to Search for Distance and Shortest Routes on Travel Salesman Problems

Paryati

UPN "Veteran" Yogyakarta,
Country Indonesia,
Email: yaya_upn_cute@yahoo.com

Abstract: The travel salesman problem is a combinatorial optimization problem that is very well-known in graph theory. The travel salesman problem is categorized as a difficult problem when viewed from a computational point of view. Also includes the classic "NP-Complete" problem because it has been studied for decades. TSP can be viewed as a matter of finding the shortest route that must be taken by someone who departs from his hometown to visit each city exactly once and then returns to his hometown of departure. In the travel salesman problem, the colony can coordinate through a very simple interaction, through this interaction, the colony is known to be able to solve very difficult problems. So, the method used to solve this TSP problem, using the Ant System algorithm is modified to the Ant Colony System Algorithm, to improve its performance on larger TSP problems. The main principle used in the AS algorithm is still used in the Ant Colony System algorithm, namely the use of positive feedback through the use of pheromones. A pheromone placed along the route is intended, so that the ants are more interested in taking that route. So that the best solution later, has a high concentration of pheromones. In order not to get trapped in the local optimal, negative feedback is used in the form of pheromone evaporation. While the main differences between the Ant System and Ant Colony System algorithms are different state transition rules, different global pheromone renewal rules, and the addition of local pheromone renewal rules. With this modification, the optimization results on the TSP obtained will be better, and get the shortest route in the minimum possible time. Based on the results of the system trials that have been carried out, it shows that the ant algorithm, both Ant Colony System and Ant System can be applied to the Travel Salesmen Problem. The Ant Colony System algorithm still has a faster search time than the Ant System algorithm and the difference is quite large.

Keywords: Ant Colony System Algorithm, Ant System Algorithm, Travel Salesman Problem, Optimization.

Nomenclature

Acronym	Description
AntCO	Ant Colony Optimization
TSP	Travelling Salesman Problem
ACS	Ant Colony System
VRP	Vehicle Routing Problem
DAACO	Dynamic Adaptive Ant Colony Optimization
GA	Genetic Algorithm
AS	Ant System

1. Introduction

In the travel salesman problem, the colony can coordinate through a very simple interaction, through this interaction, the colony is known to be able to solve very difficult problems. This is what underlies scientists to research the intelligence of insects (Swarm Intelligence). This research shows that when searching for food, ant colonies can find the shortest route from the nest to the food source. This can be done because ants communicate indirectly with other ants using pheromone trails, to choose the shortest route with high pheromone concentrations. So, in 1991, Marco Dorigo proposed a heuristic search method called the Ant System Algorithm. This method mimics the behavior of an ant colony in search of food, where a group of relatively simple artificial ants work together and communicate using pheromones on difficult discrete optimization problems. Due to the simplicity and similarity of the formula, the Ant System algorithm was first developed for the Traveling Salesman Problem.

The traveling Salesman Problem is seen as a problem of finding the shortest route, which must be taken by someone who departs from the city of origin, to visit each city exactly once, then returns to the city of origin of his departure. Traveling Salesman Problem is a problem to find Hamilton Circuit with minimum weight in a connected graph. The Traveling Salesman Problem is one of the optimization problems, which is difficult from a computational point of view. Using an exhaustive search algorithm on a complete graph. The traveling Salesman Problem can be solved by enumerating (n-1)! /2 Hamilton circuits. As a result, the computation time will increase exponentially.

The Ant System algorithm gives promising results for a small Traveling Salesman Problem. However, as the size of the Traveling Salesman Problem increases, the quality of the time and the resulting solution will decrease. To overcome all such drawbacks and limitations encountered in the former method, it is necessary to develop a reliable shortest route in the shortest time.

The objective of this paper is as follows.

1. To obtain an optimal solution in minimum time.
2. To obtain a better solution with the same number of ants and iterations.
3. To solve TSP problems in a variety of variations and applications

Then the Ant System algorithm was modified to the Ant Colony System Algorithm to improve its performance on larger Traveling Salesman Problem problems. The main principle used in the Ant System algorithm is still used in the Ant Colony System algorithm, namely the use of positive feedback through the use of pheromones. A pheromone placed along the route is intended, so that the ants are more interested in taking that route. So that the best solution later, has a high concentration of pheromones. In order not to get trapped in the local optimal, negative feedback is used in the form of pheromone evaporation. While the main differences between the Ant System and Ant Colony System algorithms are different state transition rules, different global pheromone renewal rules, and the addition of local pheromone renewal rules. With this modification, the optimization results on the Traveling Salesman Problem obtained will be better, and get the shortest route in the minimum possible time. Based on the results of system trials in this journal that have been carried out, it shows that the ant algorithm, both the Ant Colony System and the Ant System, are very suitable to be applied to the Travel Salesmen Problem. It turns out that the Ant Colony System Algorithm still has a faster search time than the Ant System algorithm and the difference is quite large.

The organization of this paper is in this order: Section 2 presents the literature review, and Section 3 portrays the basic theory. The methodology is illustrated in section 4. Section 5 covers the discussion section 6 provides the test result. Section 7 explains the advantages and disadvantages, and Section 8 concludes the paper.

2. Literature Review

Lots of journals and proceedings discuss the optimization and algorithms of the ant colony system as well as the travel salesman problem. Among other things, the Ant Colony System Optimization Algorithm that had been used so far to find the shortest route in the search for tourist sites.

In 2022, Daniel and Serly, [20] have implemented the AntCO algorithm and PHP to identify the shortest route in Palembang city between the tourist places in a website-based information system. This method solved TSP and calculated the shortest route. The input includes the number of iterations, ants, pheromone value, alpha, beta, and evaporation rate. The output provided the shortest route value between different tourist locations in Palembang City.

In 2023, Liu, H., *et al.,* [19] have executed DAACO. Initially, the parameters were initialized. Based on the pheromone trail and heuristic information the next city to be visited was identified and updated based on the quality of the solutions constructed by the ants. Then ants were divided into global search and local search. After that, A hybrid local selection strategy was used to improve the quality of ant optimization and reduce the optimization time. Finally, the algorithm terminated a satisfactory solution.

In 1999, Bullnheimer, B *et al.*, [18] have introduced an ant system algorithm for the VRP with one central depot. It contained two basic phases they are the construction of vehicle routes and trail update. The algorithm involved generating a new solution for all ants using a formula and candidate lists, improving all vehicle routes using the 2-opt-heuristic, and updating the pheromone trails using a formula. The initial placement of the artificial ants involves placing each ant for a customer to begin an iteration. Finally, the results from the computation were reported and a comparison with other metaheuristic approaches to solve vehicle routing problems is made.

In 1996, Dorigo. M, *et al.* [17] have discussed an autocatalytic process called the differential path length effect for combinatorial optimization problems. The process exploits positive feedback and involves making decisions at different times that increase the probability of making the same decision in the future. The ants exploit this process to solve optimization problems. The problem representation of the optimization problem involves a finite set of components.

In 2023, de Castro Pereira, et al., [16] have ensembled the ACO-BmTSP algorithm for solving the multiple traveling salesman problem. The ants were used to construct solutions to the mTSP. The algorithm works by iteratively constructing solutions using a probabilistic rule that was based on the pheromone trails left by the ants. The pheromone trails were updated based on the quality of the solutions found. The algorithm terminates when a stopping criterion was met, such as a maximum number of iterations or a time limit. They used a combination of two pheromone trails, one for the total travel cost and one for the balance of the tours, to construct balanced tours for the salesmen.

In 2023, Ahmed, Z.H., *et al.* [14] have modified the ant colony system algorithm for the use of global pheromone update and new heuristic information, and pheromone evaporation coefficients are used in the search space of the problem as diversification. Additionally, a 3-opt local search was used as an intensification mechanism for more quality. The algorithm was assessed on several standard problem instances, and the results showed the quality solutions with a small gap of 1% between the obtained solution and the optimal solution. The ACS algorithm was competitive with other algorithms in terms of quality.

In 2022, H.M. Asih *et al.,* [13] have solved the distribution problem for a bakery using a TSP model. This model was used to find the optimum sequence delivery nodes on tour for classical and weighted TSP. The GA method was employed to minimize the total distance traveled and thereby reduce transportation costs.

In 2023, Apostolos Tsagaris *et al.,* [10] have applied a hybrid algorithm that combines the GA and ant colony algorithm for path optimization. The ant colony algorithm was used to eliminate the parameterization uncertainty of the traditional genetic algorithm. The hybrid algorithm optimized the movement from point to point, reducing the total distance and improving the travel time. The genetic algorithm was fed by the initial population produced by the ant colony algorithm, allowing GA enrichment to produce the best results. The method was tested on a TSP problem and has applications in spatial mechanics systems such as CNC machining, robotic systems, and Coordinate Measuring Machines.

2.1 Review

Table 1 portrays the methodology, advantages, and disadvantages of the existing method. We considered eight papers that used a different methodology that used Ant Colony System Optimization Algorithm at various levels. Each method has certain benefits and shortcomings, that were explained in detail.

Table 1: *Review Based on Existing Methods*

Author	Methodology	Advantage	Disadvantage
Daniel & Serly, [20]	ACS and PHP	• Had found the shortest route for tourist sites.	• Takes a long time to process the shortest route. • Relies on Accuracy.
Liu, H., *et al.* [19]	DAACO	• Faster convergence time, better solution quality, and average value. • Achieved a higher number of optimal values on the TSPLIB dataset. • Prevent ants from falling into local optima.	• Performance depends on the problem instance and the specific parameters used. • Complexity increased with larger problem instances, which affects scalability.
Bullnheimer, B *et al.* [18]	Improved ACS	• Provide continuous solutions for hard combinational optimization problems. • Provide a high-quality solution.	• May not be efficient for very large-scale algorithms. • Didn't perform on real-world VRP instances.
Dorigo. M, *et al.* [17]	Differential path length effect	• Decentralized, scalable, robust, adaptive, and flexible. • Didn't require a central controller. • Can handle large-scale optimization problems. • Handle noisy and incomplete information. • Improve itself from previous experience. • Applied to a wide range of combinatorial optimization problems.	• Didn't provide optimal solutions for highly complex problems. • The performance depends on the specific problem instance and the parameter settings.
de Castro Pereira, *et al.* [16]	ACO-BmTSP	• Solved the multiple traveling salesman problem. • found good solutions to optimization	• It didn't guarantee the best solution. • May not be suitable for very large-scale problems due to its computational

		problems. • Provide better results for several problems.	complexity.
Ahmed, Z.H., *et al.* [14]	ACS	• Quality solutions • Competitive with other algorithms in terms of quality.	• May not be the fastest or most scalable solution for very large TSP instances. • Time-consuming • Required expert knowledge.
H.M. Asih *et al.* [13]	GA	• Provides more efficient and effective solutions to the distribution problem. • Helped to decide the product's sequence delivery nodes.	• Effectiveness depends on the specific context and variables involved in the distribution problem. • Required more computational resources and time for an optimal solution.
Apostolos Tsagaris *et al.* [10]	GA and ant colony algorithm	• Better optimization of path planning. • Eliminates the parameterization uncertainty of the traditional genetic algorithm. • Best results.	• Required more computational resources • May not be suitable for all types of optimization problems. • Required customization for specific applications.

2.2 Research Gaps

Some of the common issues faced in TSP are as follows,

The papers [20][16][14][13][10] provided better results to find the shortest path to reach the exert destination. However, these methods required high computational resources and time to process during iteration to get accurate results. In [19][17], used to work faster and provide better results for the shorter TSP. It failed to provide the same fast and best result when the TSP was complex.

3. Basic Theory

3.1 Graph

Graphs are used to represent discrete objects and the relationships between these objects. The graph, $G = (V, E)$, which in this case:

V = non-empty set of vertices = $\{ v1 , v2 , \ldots , vn \}$

E = set of edges connecting a pair of vertices = $\{e1 , e2 , \ldots , en \}$

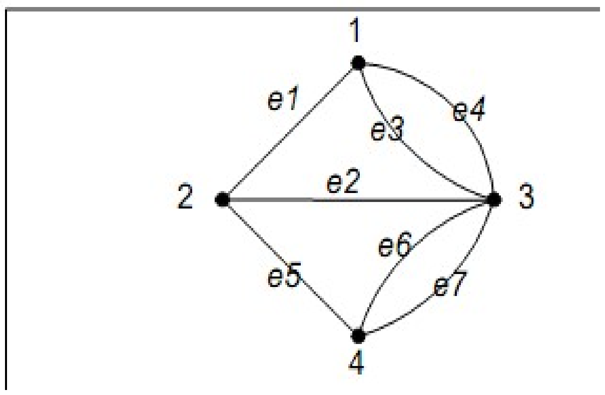


Fig.1. *Example Graph*

G is a graph with

$V = \{ 1, 2, 3, 4 \}$

$E = \{ (1, 2), (2, 3), (1, 3), (1, 3), (2, 4), (3, 4), (3, 4) \}$

$= \{ e1, e2, e3, e4, e5, e6, e7\}$

3.1.1 Types of Graphs

Based on the presence or absence of loop or multiple/parallel edges in a graph, graphs can be grouped into two, namely:

a. Simple graph (simple graph)

It is a graph that has neither ring nor double edges, as shown in Fig 2-a. The side of the bracelet is the side that starts and ends at the same vertex, while the double side is the side that connects the two same vertices.

b. Un-simple graph

It is a graph that has circular or double edges, there are two types of non-simple graphs, namely pseudographs (Fig 2-b) and double graphs (multigraph) (Fig 2-c). Pseudographs are more common than double graphs because the edges of a pseudo graph can be connected to themselves (forming a ring). Based on the orientation of the direction on the sides, the graph is divided into two types, namely:

a. Undirected graph (undirected graph)

A graph whose edges have no direction orientation is called an undirected graph. (Fig 2(a,b,c))

b. Directed graph (or digraph)

A graph in which each edge is oriented in a direction is called a directed graph (Fig 2-d).

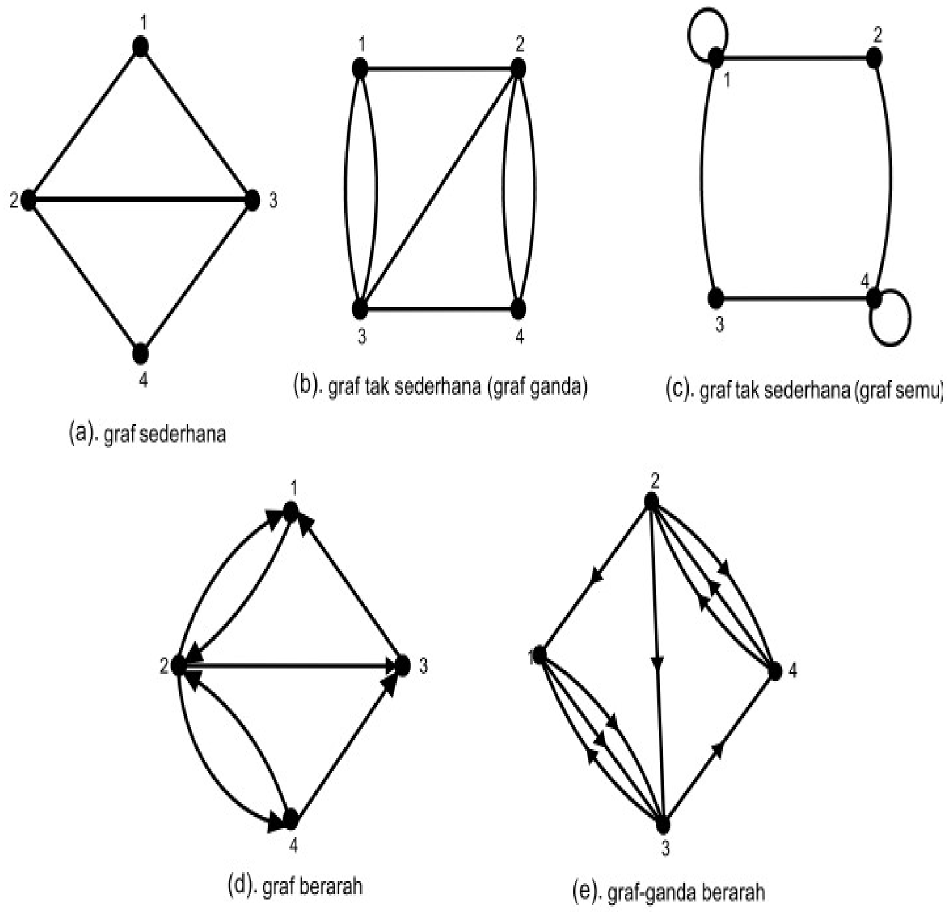

(a). graf sederhana
(b). graf tak sederhana (graf ganda)
(c). graf tak sederhana (graf semu)
(d). graf berarah
(e). graf-ganda berarah
1
2
3
4

Fig.2. *Types of Graphs*

The definition of the graph above can be extended to include a directed double graph (Fig 2-e). In a directed double graph, loops, and double edges are allowed. A summary of the definition expansion can be seen in Table 2 below.

Table 2: *Summary of Graph Types*

Type	Side	Double Side Allowed?	Side Bracelet Allowed?
Simple graph	Undirected	No	No
Undirected	Doublegrah	Yes	Yes
Pseudograph	Undirected	Yes	Yes
Directed	Graph	No	Yes
Directed	Double-directed graph	Yes	Yes

3.1.2 Hamilton Tracks and Circuits

It is defined as a path that passes through each vertex in the graph exactly once. If the path returns to the origin so that it forms a closed path, then the closed path is called a Hamilton Circuit. In other words, a Hamilton circuit is a circuit that passes through each vertex in the graph exactly once, except for the start vertex (and the end vertex) which is traversed twice. A graph that has a Hamilton circuit is called a Hamiltonian graph, while a graph that only has a Hamiltonian path is called a semi-Hamilton graph (Fig 3). Every complete graph is a Hamiltonian graph.

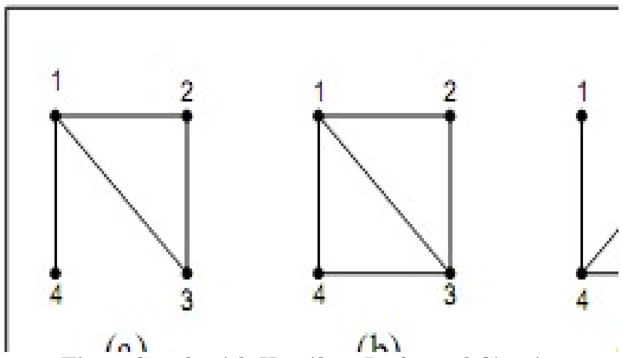


Fig.3. *Graph with Hamilton Paths and Circuits*

3.2 N-Complete Problem

It has a simple problem formulation. For small inputs, the "NP-Complete" problem can be solved quickly. But if the input gets bigger, then the time takes will increase tremendously (usually increases exponentially). The properties possessed by "NP-Complete" are as follows:

a. Very difficult to solve on the computational side. No algorithm can solve it in the order of polynomial time.
b. There is no evidence that an NP-Complete problem cannot be solved in the order of polynomial time.
c. If any "NP-Complete" problems can be solved in the order of polynomial time, then all problems that are included in the "NP-Complete" can also be solved in the order of polynomial time.
d. On the other hand, if any "NP-Complete" problem can be proven to be unsolvable in the order of polynomial time, then all problems that are included in the "NP-Complete" are also proven to be unsolvable in the order of polynomial time.

If the optimal solution to the "NP-Complete" problem cannot be obtained efficiently, then in practice optimization is often sacrificed for the sake of efficiency [7]. In other words, instead of looking for an optimal solution, it is better to find a solution that is close to optimal as long as it is completed in the order of polynomial time. This is done by using a heuristic method. The heuristic method aims to find a very good solution (near optimal) to a combinatorial optimization problem [11], although it does not guarantee an optimal solution. However, due to lower computational costs, most of the "NP-Complete" problems are solved using heuristic methods.

3.3 Traveling Salesman Problem

TSP is a combinatorial optimization problem that is very well-known in graph theory. TSP is categorized as a difficult problem when viewed from a computational point of view. Also includes the classic "NP-Complete" problem because it has been studied for decades. TSP can be viewed as a matter of finding the shortest route that must be taken by someone who departs from his hometown to visit each city exactly once and then returns to his hometown of departure. In graph theory, TSP is expressed as a graph $GTSP = (N, A)$ that N represents the city vertices, while A is the set of edges connecting the city vertices in the graph. The weights on the sides represent the distance between the two cities. So, TSP is nothing but a problem to find a Hamilton Circuit that has a minimum weight on a connected graph. For the TSP of any complete graph, the calculation becomes $(n-1)!/2$ Hamilton Circuits. However, for large n calculations, the number of Hamilton Circuits that must be examined one by one will be very large. For $n = 20$ there will be $6\ x\ 1016$ solutions. As a result, the computation time required will also increase exponentially (the complexity is $O(n!)$). In other words, TSP belongs to the category of NP-Complete problems.

3.4 Ant Colony Behavior When Looking for Food

When looking for food, ants' knowledge of their environment is very limited. Individually, ants will find it difficult to find a route to a food source if the search is carried out over a large area. To overcome this deficiency, the ants will work collectively in the colony. The concentration routes of ants in foraging for food in the colony are,

a. At first, the concentration of pheromones increased because the route passed was a shorter route (the process of going back and forth was faster).
b. High pheromone concentrations will attract the attention of other ants.
c. As a result, after a while, the concentration of pheromones becomes higher because more ants pass through that route than other routes.

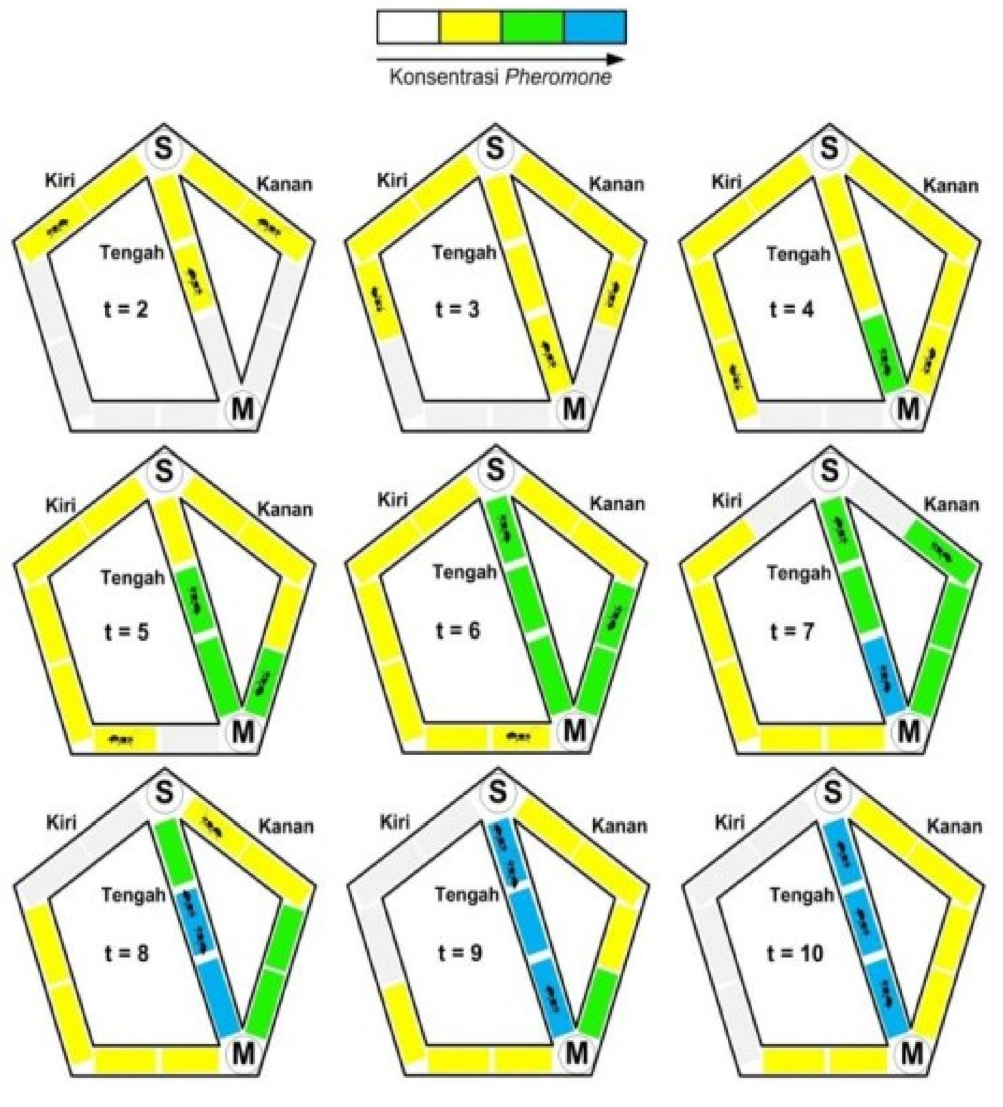


Fig.4. *The process of searching for food by an ant colony*

This behavior is better known as positive feedback using pheromones. The higher the accumulation of pheromones on a route, the more interesting the route is to follow. In other words, the probability of a route being chosen is directly proportional to the concentration of pheromone on that route. This behavior underlies the optimization process in the ant algorithm.

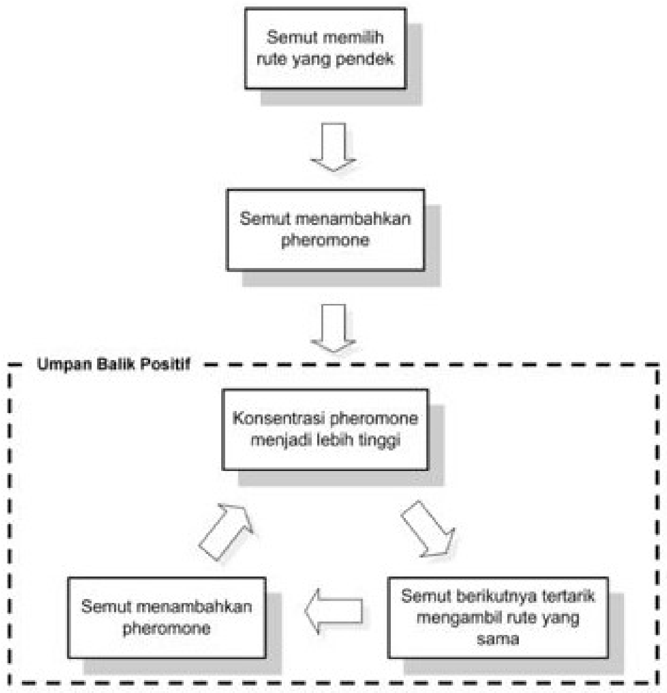

Semut memilih rute yang pendek
Semut menambahkan pheromone
Umpan Balik Positif
Konsentrasi pheromone menjadi lebih tinggi
Semut menambahkan pheromone
Semut berikutnya tertarik mengambil rute yang sama

Fig.5. *Feedback Ant Colony*

3.5 Ant Algorithm

The ant algorithm was proposed by Marco Dorigo in 1991 and was inspired directly by observations of ant colonies. In this algorithm, colonies of "artificial ants" will work together to find solutions to discrete optimization problems while communicating indirectly with pheromones. The final solution given is the result of collective work within the colony.

3.5.1 Characteristics of Ant Algorithm

The ant algorithm is constructive because each "artificial ant" (hereinafter referred to as "ant") will build a solution in stages. As an analogy, the ant colony will traverse the exploratory graph $GS = (C, L)$ were,

a. C is a set of candidate solution nodes.
b. L is the set of edges that expresses the relationship between pairs of vertices C. The pair (ci, cj) (ci, cj) can be an indication that the pair is a feasible solution (feasible).
c. The weights on the side (ci,cj) represent the costs required to make the pair (ci,cj) a feasible solution.

Each k ant in the ant algorithm will have the following characteristics:

a. Ants will explore the graph GS trying to find the optimal solution.
b. Ants have Mk memory which stores information about the route taken so far. This information can be used to build and evaluate solutions.
c. Before starting the search, each ant has certain initial conditions. Generally, the initial conditions for each ant are expressed by the empty solution set.
d. Each ant will visit each node according to the problem constraints while building the solution step by step. The search stops when there are no more feasible nodes to add or it has reached a certain stop condition.
e. Ants move probabilities using certain state transition rules (State transition rule). This rule is a function of the pheromone concentration, heuristic information, and the memory of the ant concerned.
f. At the end of each iteration, a renewal process is carried out in the form of adding pheromones by ants and evaporation of pheromones on the GS side.

Informally, the ant algorithm consists of 3 main parts: initialization, solution construction, and pheromone update.

```
Procedure AlgoritmaSemut
  Inisialisasi
  While (kondisi berhenti belum dipenuhi) do
    BangunSeluruhSolusi
    PerbaharuiPheromone
  End
End
```

From the data above, it can be seen that the solution development process and pheromone renewal will continue to be repeated until the stopping conditions are met.

3.5.2 Initialization

Initialization is done so that each ant is ready to explore the graph. In general, the steps taken at the time of initialization are:

a. Choose the number of ants to use
b. Place each ant on its vertex at random.
c. Empty/initialize the memory of each ant.
d. Initialize pheromones and heuristic information.

3.5.3 Solution Development

The ant colony will build the solution step by step by visiting all the nodes according to the problem constraints. The vertices to be visited are selected based on certain transition rules by considering pheromone factors, heuristic information, and ant memory. Pheromones can be thought of as another form of colony experience during the process of laxatives. At first, the ants do not have experience because all edges of the graph have the same pheromone. After a while, the experience of the ants will increase along with the difference in pheromones that arise as a result of the colony's activity during their search Heuristic information is a value outside of the pheromone that can give the ant an idea about the graph being explored. In general, this information is very useful at the beginning of the search when the ants are still inexperienced (there is no difference in pheromones). Because the heuristics are initialized from the start, this information can guide the ant from being completely “blind” when exploring a new area. As experience increases, the influence of this information will decrease.

3.5.4 Pheromone Update

The pheromone renewal process is a modification process to the pheromone in the exploration graph. Pheromones can increase due to the addition of ants or decrease due to evaporation. Put simply, positive feedback through the process of adding pheromones will increase the chances of a side being re-elected in the future. On the other hand, pheromone evaporation is a form of negative feedback that is useful so that ants can "forget" their previous choice and try to find alternative routes so they are not trapped in the local optimal.

4. Method

4.1 Ant Algorithm on TSP

Ant algorithm was developed to solve various discrete optimization problems. Ant algorithm will be applied to the TSP problem. This problem was chosen because

a. The behavior of the ant colony when foraging is similar to the TSP problem formulation.
b. TSP is an NP-Complete discrete optimization problem.
c. TSP is the most studied discrete optimization problem, so it is often referred to as the benchmark problem.
d. The simplicity of the TSP formulation.
e. TSP can be developed into a variety of variations and applications.

Meanwhile, the TSP used in the later test is a completely connected, undirected graph with no more than 550 vertices.

4.1.1 Graph Exploration

The graph used by ants to explore is identical to the problem graph in the TSP. the exploratory graph component $GS = (C, L)$ corresponds to the TSP graph $GTSP = (N, A)$. The components of the node set C in GS correspond to the city set N in $GTSP$. The weight on the edge set L corresponds to d_{ij} which represents the distance between city i to city j.

4.1.2 Pheromones and Heuristic Information

The pheromone placed on the TSP expressed the ants' interest in visiting the city j from the city i. The pheromone expressed ij will be placed along the side of the road between cities i and j. The initial value of the pheromone $(\tau 0)$ is set so that it is slightly larger than an ant can put in one iteration. This is important because if the value 0 is too small, the search will be one-sided. After all, ants tend to always choose the route that has been found previously. Meanwhile, if the value 0 is too large, then the added pheromone will have no meaning during the search process. Heuristic information is expressed by ij. For TSP, the heuristic information will state the visibility of the ants to the surrounding cities. The value of the visibility of the ant to the surrounding cities. The visibility value ij is calculated heuristically based on the equation below.

$$\eta_{ij} = \frac{1}{d_{ij}}$$

The above equation states that visibility ij is a constant value because it represents the inverse between the distances of cities i and j (d_{ij}). Information on visibility will be very useful at the beginning of the search when the ants do not have experience (because in TSP there is no difference in pheromone concentrations).

4.1.3 Solution Development

The only limitation of TSP is that each city must be visited once (except the starting city). This constraint is implemented by requiring each ant, to ensure that the city to be visited has never been visited before. This is done through the ant's Mk memory. Each ant in the colony will build its solution. Every ant has an arbitrary starting city that is different from one another. At each step, it will iteratively add the visited city to the tour list. This process will be completed when all the ants have completed their tour.

4.2 AS

The AS algorithm is the first attempt to apply ant colony behavior to TSP. There are three versions of the AS Algorithm developed: ant-density, ant-quantity, and ant-cycle. The difference between the three lies in how to calculate the pheromone to be added. In the ant-density and ant-quantity versions, the added pheromone is calculated at each step of the ant's movement to the new city. Meanwhile, in the ant-cycle version, the pheromone to be added is calculated after the ants have completed their respective tours because the performance of ant-cycle is better than ant-density or ant-quantity, so when referring to the AS algorithm, it is the ant-cycle version. The pseudo-code of the AS algorithm can be seen below.

```
inisialisasi
while (not kondisi berhenti) do
    pembangunan solusi
    pembaharuan pheromone global
end while
```

4.2.1 Initialization

Initially, the pheromone $(\tau 0)$ is initialized on each side of the TSP using a heuristic approach. Initialization rules can be seen below.

$$\tau_{ij} = \tau_0 = \frac{m}{C_{greedy}}$$

Where m represents the number of ants and C represents the length of the tour generated through the usual greedy algorithm. After that, the heuristic information is initialized in the form of visibility (η_{ij}) which is calculated based on the previous equation.

4.2.2 Solution Development

The highest number of ants used in the US Algorithm should be the same as the number of cities in the TSP. After each ant is in the city i, it will choose the city j according to the proportional random state transition rule. In this rule, ants will choose a city at random but tend to choose the city with the greatest probability. This rule is stated as follows:

$$p_{ij}^{k} = \begin{cases} \dfrac{[\tau_{ij}]^{\alpha}[\eta_{ij}]^{\beta}}{\sum\limits_{j \in N_i^k} [\tau_{ij}]^{\alpha}[\eta_{ij}]^{\beta}}, & \text{jika}\, j \in N_i^k \\ 0, & \text{sebaliknya} \end{cases}$$

Where $p^{k}{}_{ij}$ represents the probability of an ant k visiting the city j from the city i. ij states the antler's visibility of the cities around i, N_i^k is the set of neighbors of the city i that may be visited by k ants. while and are constants that express the strong influence of pheromone $(\tau 0)$ and visibility (η_{ij}) on the choice of ants. If the constant = 0, the tendency of the nearest or similar city to the greedy search process. If $p = 0$, then the search only relies on pheromones, which will usually lead to poor search results. Later all the cities visited by the ants will be recorded in the Mk memory. Apart from ensuring that ants do not visit the same city, this memory is also used to calculate the distance traveled. The AS algorithm can be implemented in parallel or sequentially. In a parallel search, all ants move from one city to another (constructing a solution) congruently. Whereas in sequential search, a solution must be built first, before the next solution can be built by other ants.

4.2.3 Global Pheromone Update

Once all the ants have completed their tour, the pheromone will be completely updated. This will be updated completely. This is done by first evaporating the pheromone over the entire TSP using the following equation.

$$\tau_{ij} \leftarrow (1-\rho)\tau_{ij}, \quad \forall (i,j) \in TSP$$

Where ($0 < 1$) is a constant such that the value ($1 - p$) will describe the rate of evaporation of the pheromone. After evaporation, TSP will receive many pheromones based on the equation below.

$$\tau_{ij} \leftarrow \tau_{ij} + \sum_{k=1}^{m} \Delta\tau_{ij}^{k}, \quad \forall (i,j) \in TSP$$

Where ij represents the new pheromone level and represents the amount of pheromone to be added by the ant k. The value k_{ij} is calculated based on the following equation.

$$\tau_{ij} \leftarrow (1-\rho)\tau_{ij}, \quad \forall (i,j) \in TSP$$

Where Ck states the length of the Tk tour. While the Tk tour contains the TSP side taken by k ants. Based on the last equation, it can be concluded that the shorter the route chosen, the greater the amount of pheromone added. If the pheromone level is higher, then the route is likely to be re-selected in the next iteration. Through testing, the AS Algorithm gives promising results at a small TSP $(n \leq 30)$. However, the performance of the AS algorithm tends to decrease with increasing input size. Therefore the AS Algorithm was modified to be the ACS Algorithm.

4.3 ACS

ACS is a development of the AS Algorithm. Broadly speaking, the ACS Algorithm is the same as AS. However, it has 3 main differences, namely:

a. Changes in state transition rules that try to balance the tendency to explore old routes with the desire to explore new routes.
b. Global pheromone updates (evaporation and additions are only performed on the best routes at this time.
c. Added local pheromone update rule every time an ant moves to the next town.

The Pseudocode of the ACS Algorithm can be seen below.

```
Initialization
While (not stop condition)
do
While (solution development)
do Visit the next city Local pheromone
update End
while Local pheromone
update End
while
```

4.3.1 Initialization

The initial initialization of the pheromone $(\tau 0)$ in the ACS Algorithm is determined heuristically through the equation:

$$\tau_{ij} = \tau_0 = \frac{1}{nC_{greedy}}$$

Where n is the number of cities in the TSP, while C_{greedy} states the best route based on the greedy algorithm. As for the visibility initialization (η_{ij}), the rules used are the same as in the AS Algorithm.

4.3.2 Solution Development

The state transition rules used in the ACS Algorithm are different from the AS Algorithm. This rule is called pseudorandom proportion which is expressed through the equation.

$$j = \begin{cases} \arg\max_{l \in N_i^k} \{\tau_{il}[\eta_{il}]^{\beta}\}, & \text{jika } q \le q_0 \text{ (eksploitasi)} \\ J, & \text{jika } q > q_0 \text{ (eksplorasi)} \end{cases}$$

Where q is a random number with a value between 0 to 1, q_0 is a constant that states the strength of the visibility effect (η_{ij}), N_i^k is a collection of cities around i chosen by k ants. while J the city chosen based on the equation.

$$p_{ij}^k = \begin{cases} \dfrac{[\tau_{ij}][\eta_{ij}]^{\beta}}{\sum_{j \in N_i^k} [\tau_{ij}][\eta_{ij}]^{\beta}}, & \text{jika } j \in N_i^k \\ 0, & \text{sebaliknya} \end{cases}$$

The pseudorandom proportional rule works as follows:

a. First, a random number q is generated.
b. If $q\ q_0$, then the ants will choose the city with the best value based on pheromone information (τ_{ij}) and visibility (η_{ij}) of all cities that may be visited.
c. If $q > q_0$, then the ant will choose the city based on the above equation.

From the information above, it can be concluded that at probability q_0, ants will choose the side of the road with pheromone values and good visibility (exploring information that is considered good).

While the probability $(1 - q_0)$ ants will explore a new route. So, by setting the q_0 parameter, it is expected to balance the tendency of ants to take the old route (considered good) with the desire to explore alternative routes.

4.3.3 Global Pheromone Update

The global pheromone update process in the ACS Algorithm is only carried out on the best route at this time. Just like the AS Algorithm, this process is done at the end of each iteration after all the ants have finished their tour. The pheromone renewal rule in ACS is given in the equation:

$$\tau_{ij} \leftarrow (1-\rho)\tau_{ij} + \rho\Delta\tau_{ij}^{best}, \quad \forall(i, j) \in T^{best}$$

The best ij is calculated based on the equation:

$$\Delta\tau_{ij}^{best} = 1 / C^{best}$$

The above equation states that evaporation and addition of pheromone are only carried out on the best route (τ^{best}), Not all the sides of tsp. same AS Algorithm, the value $(1 - p)$ also represents the evaporation of the pheromone.

4.3.4 Local Pheromone Update

There is a new rule in the ACS Algorithm. This rule states that an ant moves from city i to city j then the pheromone on the (i, j) side before completing its tour [7]. Local pheromone updates are expressed in the equation below:

$$\tau_{ij} \leftarrow (1-\xi)\tau_{ij} + \xi\tau_0$$

The effect of local pheromone renewal is that every time an ant passes an edge (i, j), the concentration of pheromone on that side will decrease. As a result, in the next iteration, sides (i, j) become less attractive to visit. In other words, this rule can make the attractiveness of a side decrease dynamically. It will increase the ants' desire to explore alternative solutions. Without this rule, ants will tend to explore routes that have previously been found (because they are considered good) [5]. Previously it was mentioned that the search on the AS Algorithm can be done in two ways: parallel or sequential. Unlike the AS algorithm, because the ACS algorithm has local pheromone updating rules, the search on the ACS algorithm should be done in parallel.

5. Discussion

5.1 Software Requirements

The software uses the Java 6 programming language with NetBeans IDE 6.5 as follows:

a. The software is capable of opening archives containing TSP descriptions.
b. The software can find a solution on the TSP by using the AS and ACS Algorithms.
c. The software provides facilities to set the algorithm and constants to be used.
d. The software can find solutions from TSP using AS or ACS algorithms and display the process visually.

5.2 Use Case Diagram

Based on the specifications of the software requirements created, the identified use cases are as follows:

a. **Use Case "Opening TSP",** is the functionality used by the user/user to open the TSP archive to be completed. The archive contains the name of the problem, comments, the number of cities, the type of problem, and the optimal solution.
b. **Use Case "Creating TSP",** to create your own TSP.
c. **Use Case “Saving TSP”**, to save the created TSP.
d. **Use Case "Solving TSP",** is a functionality to solve TSP using AS or ACS Algorithm.
e. **Use Case "Setting Algorithm",** is a functionality to select the desired algorithm, constants, and stopping conditions.
f. **Use Case "Calculating Greedy"**, is a functionality to find a new solution for TSP based on a greedy algorithm.

Fig 6 is the use Case Diagram of the software.

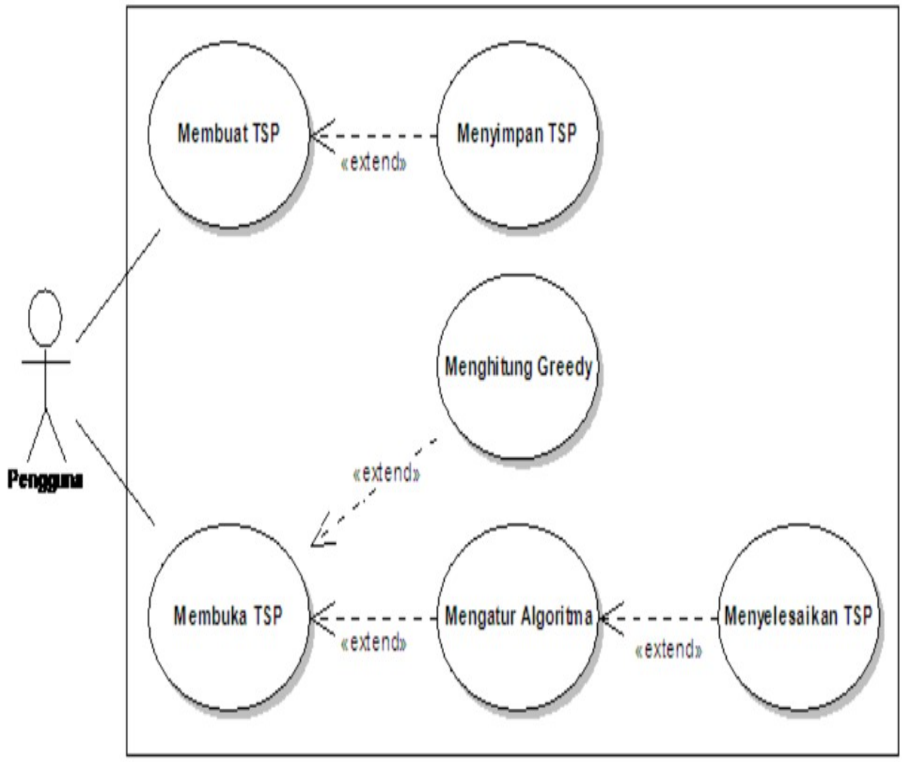


Fig.6. *Software Use Case*

5.3 Class Implementation

The required classes are implemented using the Java 6 programming language with Netbeans IDE 6.5. The implemented classes can be seen in the following Table 3.

Table 3: *Class Implementation*

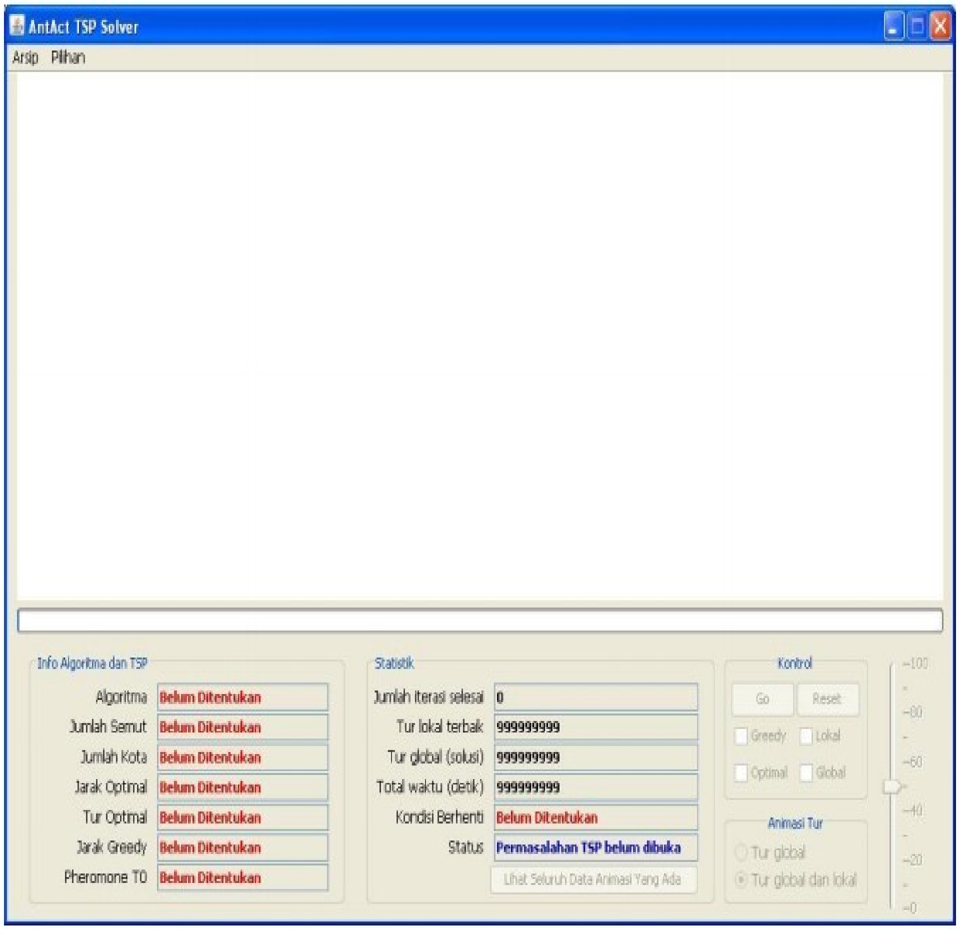
No	Nama Kelas	Nama File
1.	AntActGUI	AntActGUI.java
2.	DialogAturAlgoritma	DialogAturAlgoritma.java
3.	DialogBuatTSP	DialogBuatTSP.java
4.	DialogBukaTSP	DialogBukaTSP.java
5.	DialogDataAnimasi	DialogDataAnimasi.java
6.	TSP	TSP.java
7.	Algo	Algo.java
8.	ParserTSP	ParserTSP.java
9.	DrawHandler	DrawHandler.java
10.	C_BuatTSP	C_BuatTSP.java
11.	C_Main	C_Main.java
12.	Ant	Ant.java
13.	Bound	Bound.java
14.	Matriks	Matriks.java
15.	Coordinate	Coordinate.java
16.	Tabel	Tabel.java
17.	Tour	Tour.java
18.	Utilities	Utilities.java

5.4 Implementation of the Interface

The software developed has five interfaces which are implemented into five classes. The implementation of all classes can be seen in Table 3. While the overall appearance of the interface can be seen in Fig 7 to 11.

Table 4: *Implementation of Interface Classes*

No	Antarmuka	Nama File
1.	Antarmuka utama	AntActGUI.java
2.	Antarmuka membuka arsip TSP	DialogBukaTSP.java
3.	Antarmuka membuat TSP	DialogBuatTSP.java
4.	Antarmuka untuk mengatur algoritma	DialogAturAlgoritma.java
5.	Antarmuka data animasi	DialogDataAnimasi.java



Info Algoritma dan TSP

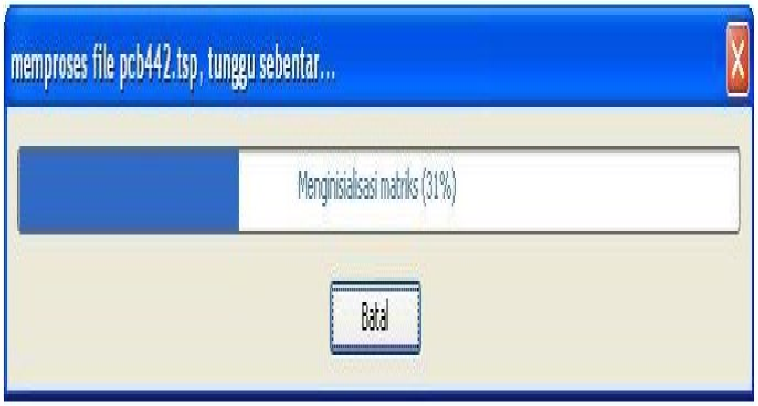
Algoritma	Belum Ditentukan
Jumlah Semut	Belum Ditentukan
Jumlah Kota	Belum Ditentukan
Jarak Optimal	Belum Ditentukan
Tur Optimal	Belum Ditentukan
Jarak Greedy	Belum Ditentukan
Pheromone T0	Belum Ditentukan

Statistik

Jumlah iterasi selesai	0
Tur lokal terbaik	999999999
Tur global (solusi)	999999999
Total waktu (detik)	999999999
Kondisi Berhenti	Belum Ditentukan
Status	Permasalahan TSP belum dibuka

Lihat Seluruh Data Animasi Yang Ada

Fig.7. *Implementation of the Main Interface*


memproses file pcb442.tsp, tunggu sebentar...

Menginisialisasi matriks (31%)

Fig.8. *Interface Implementation Opening the TSP archive*

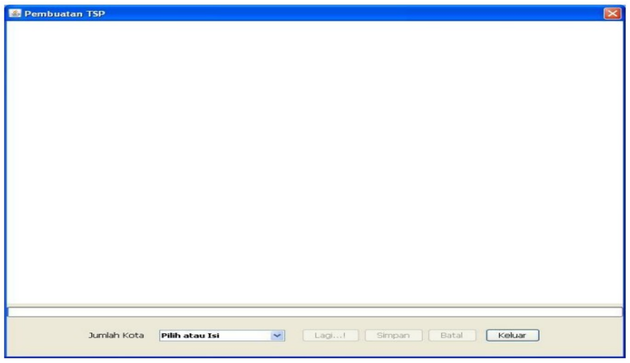
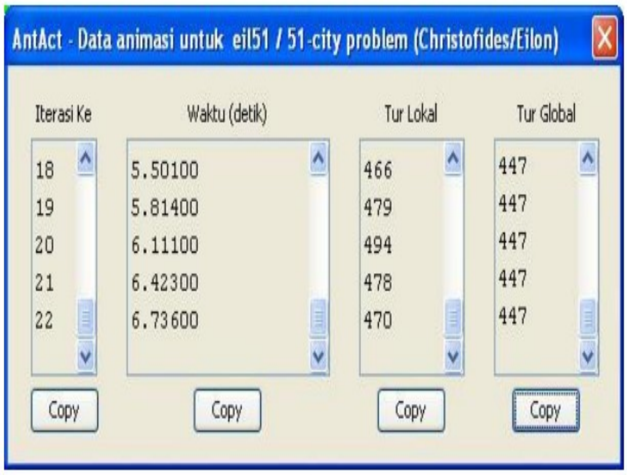
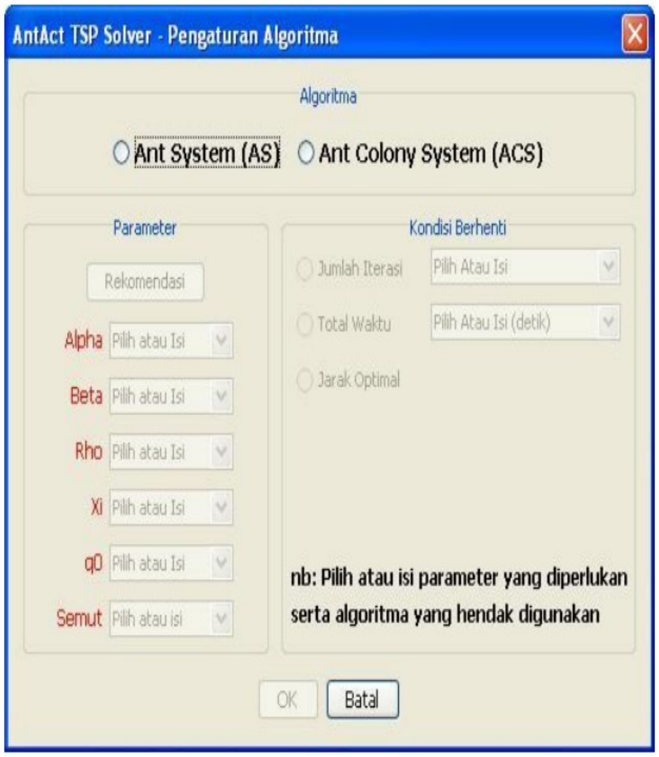


Fig.9. *Implementation of the TSP Creation Interface*


AntAct - Data animasi untuk eil51 / 51-city problem (Christofides/Eilon)

Iterasi Ke	Waktu (detik)	Tur Lokal	Tur Global
18	5.50100	466	447
19	5.81400	479	447
20	6.11100	494	447
21	6.42300	478	447
22	6.73600	470	447

Fig.10. *Animated Data Interface Implementation*


AntAct TSP Solver - Pengaturan Algoritma

Algoritma: Ant System (AS) / Ant Colony System (ACS)

Parameter: Alpha: Pilih atau Isi; Beta: Pilih atau Isi; Rho: Pilih atau Isi; Xi: Pilih atau Isi; q0: Pilih atau Isi; Semut: Pilih atau isi

Kondisi Berhenti: Jumlah Iterasi: Pilih Atau Isi; Total Waktu: Pilih Atau Isi (detik); Jarak Optimal

nb: Pilih atau isi parameter yang diperlukan serta algoritma yang hendak digunakan

Fig.11. *Interface Implementation Set Algorithm*

6. Test Result

The planned tests are as follows:

a. Testing the AS and ACS algorithms using a TSP with varying numbers of ants and iterations.
b. Testing the AS and ACS algorithms using various TSPs with the same number of ants and iterations for each algorithm.
c. The TSP used in the test was taken from TSPLIB95, which is a TSP library that is often used in research.
d. The constant parameters used in the AS and ACS Algorithms follow the suggestions, namely:
e. AS Algorithm: $= 1, \ = 2, \ = 0.5$
f. ACS Algorithm : $= 1 \left(always\right), \ = 2, \ = 0.1. \ = 0.9, \ q0 \ = \ 0.1$ 5.1
g. Testing The TSP used is in the size of 14 cities to 532 cities. The number of ants used is half the number of cities. Each TSP is tested with 500 iterations. While the number of tests on the TSP is as follows:
h. TSP measuring 14 *to* 100 cities tested 5 times.
i. TSP measuring 101 *to* 300 cities tested 3 times.
j. TSP above 300 cities tested 1 time.

The average solution will be recorded, then the deviation is calculated using the formula.

$$\text{Simpangan} = \frac{\text{Solusi Algoritma}}{\text{Solusi Optimal}} \times \text{Solusi Optimal}$$

6.1 Analysis of Deviations at Various TSP

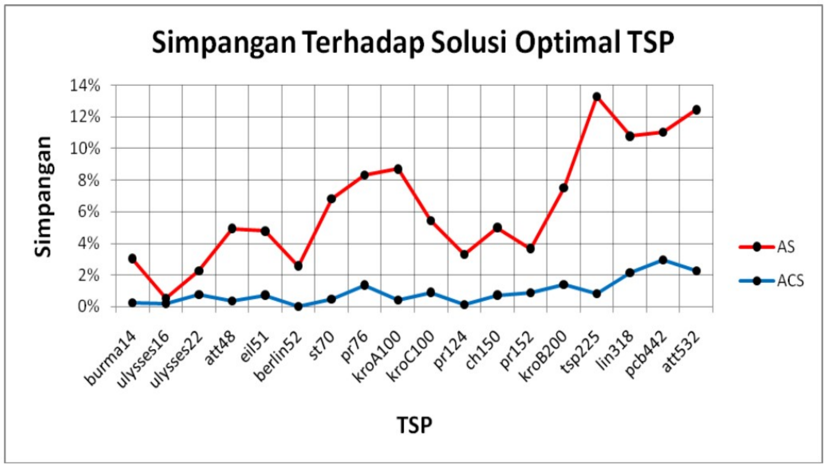


Fig.12. *Deviations of various TSPs toward the Optimal Solution*

From Fig 12, it can be seen that as the size of the TSP increases, the deviation difference between the two gets bigger. However, the deviation of the ACS Algorithm is not as large as AS. Where in the AS Algorithm, the largest deviation is in TSP225 (13.270%).

6.1 Analysis of Search Time Against Various TSP

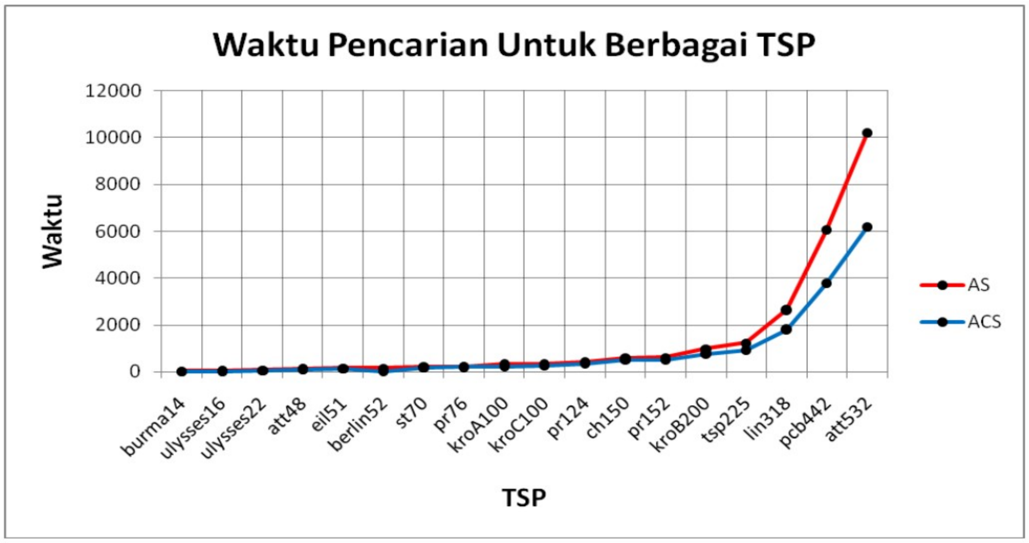


Fig.13. *Graph of Search Time Against various TSP*

Based on Fig 13, it can be seen that the larger the TSP size, the greater the search time required. Based on Table 5, for small TSP (< 100 cities) the ACS Algorithm has a faster average search time (except eil51) than the AS Algorithm although the difference is not too big. However, for larger problems (> 100 cities). The ACS algorithm still has a faster search time than the US algorithm and the difference is quite large.

Table 5: *Algorithm search time on various TSP*

No	TSP	Semut	Optimal	Waktu Pencarian	
				AS	ACS
1	burma14	7	3323	43.958	33.670
2	ulysses16	8	6859	50.047	45.122
3	ulysses22	11	7013	67.887	63.611
4	att48	24	10628	143.812	123.277
5	eil51	26	426	152.530	152.722
6	berlin52	26	7542	154.865	37.448
7	st70	35	675	208.859	208.788
8	pr76	38	108159	226.971	226.352

7. Advantages and Disadvantages

Advantages:

a. In this method, both Ant Colony System and Ant System were applied to the Travel Salesman Problem.
b. The software built can solve the Travel Salesman Problem with the specified library and display it visually.
c. Testing with TSPLIB95 showed an increase in the number of ants in the Ant system and Ant Colony System.
d. Testing with TSPLIB95 shows that with the same number of ants and the same number of iterations, the Ant Colony System Algorithm provides a better solution and a faster search time than the AS Algorithm.

Disadvantages:

a. This algorithm can improve the resulting solution, but in the Ant Colony System Algorithm, the effect of the increase is not as strong as in the Ant System Algorithm.

8. Conclusion

The conclusions obtained are as follows:

a. In Ant algorithms, both ACS and AS can be applied to TSP.
b. The software that is built can solve TSP with a predetermined library and display it visually.
c. Tests with TSPLIB95 showed an increase in the number of ants in the AS and ACS. Algorithms were able to improve the resulting solution, but in the ACS Algorithm, the effect of the increase was not as strong as in the AS Algorithm.
d. Testing with TSPLIB95 shows that with the same number of ants and the same number of iterations, the ACS Algorithm provides a better solution and a faster search time than the AS Algorithm.

Our future work will focus on evaluating the effectiveness of the path in the shortest time.

Compliance With Ethical Standards

Conflicts of interest: Authors declared that they have no conflict of interest.

Human participants: The conducted research follows ethical standards and the authors ensuredthat they have not conducted any studies with human participants or animals.

References

[1] Marco, and Dorigo., "Ant Colony System: A Cooperative Learn in Approach to the Traveling Problem", Université Libre de Bruxelles. Mike, "Visual Bots – Solving The TSP", 2019.
[2] Wikipedia Indonesia, "Ant Algorithm – Indonesian Wikipedia, a free encyclopedia in Indonesian", 2021.
[3] Wikipedia Indonesia, "Travel Salesman Problem – Wikipedia Indonesia, the free encyclopedia in Indonesian", 2020.
[4] Wikipedia Indonesia, "Ant colony optimization - Wikipedia, the free encyclopedia", 2021.
[5] Wikipedia Indonesia, "Application and Programming Netbean - Wikipedia, the free encyclopedia", 2021.
[6]http://www.iwr.uniheidelberg.de/groups/comopt/SoftwareJavaandApplication Programming/TSPLIB95/index.html/
[7] http://www2.research.att.com/~dsj/chtsp/
[8] http://www.visualbots.com/mandelbrot_project.htm
[10] Tsagaris, A., Kyratsis, P. and Mansour, G., "The Integration of Genetic and Ant Colony Algorithm in a Hybrid Approach", International Journal of Intelligent Systems and Applications in Engineering, Vol. 11, No. 2, pp.336-342, 2023.
[11] R. Astri and Sularno, "Implementation of A-Star Algorithm for Searching Routes Near the Tsunami Evacuation Shelter Point," J. RESTI (Sist. and Technol. Information), Vol. 4, No. 2, pp. 254–259, 2020.
[12] Alhanjouri, M. & Alfarra, B., "Ant Colony Versus Genetic Algorithm Based on Traveling Salesman Problem", International Journal Comp.Tech.Appl., Vol. 2, No. 3, pp.570-78, 2012.
[13] Asih, H.M., Leuveano, R.A.C., Rahman, A. and Faishal, M., "TRAVELING SALESMAN PROBLEM WITH PRIORITIZATION FOR PERISHABLE PRODUCTS IN YOGYAKARTA, INDONESIA".
[14] Ahmed, Z.H., Yousefikhoshbakht, M., Saudagar, A.K.J. and Khan, S., "Solving the Travelling Salesman Problem Using an Ant Colony System Algorithm", *IJCSNS*, Vol. 23, No.2, pp.55, 2023.
[15] M. D. Khairansyah, M. Luqman Ashari, and I. Mufidah, "Determining the Path The Shortest Evacuation In The Industry Plastic Using Ant Colony", Vol. 3, No. 1, 2022.
Optimization," J. Kelam. transp. Road (Indonesian J. Road Safety), Vol. 8, No. 1, pp. 53–61, 2021.
[16] de Castro Pereira, S., Solteiro Pires, E.J. and de Moura Oliveira, P.B., "Ant-Balanced Multiple Traveling Salesmen: ACO-BmTSP. Algorithms", Vol. 16, No. 1, pp.37, 2023.
[17] Dorigo, M., "The Ant Colont Optimization Metaheuristics: Algorithms, Applications and Advances", University of Liberty de Bruxelle, IRIDIA, 1996.
[18] Bullnheimer, B., Hartl, R. F., and Strauss, C., "An improved ant system algorithm for the vehicle routing problem, technical report, Institute of Management Sciences, University of Vienna, Austria", Vol.89, pp. 319-328, 1999.
[19] Liu, H., Lee, A., Lee, W. and Guo, P., "DAACO: adaptive dynamic quantity of ant ACO algorithm to solve the traveling salesman problem", *Complex & Intelligent Systems*, pp.1-14, 2023.
[20] Daniel U, Serly O, "Penerapan Algoritma Ant Colony Optimization Untuk Pencarian Rute Terpendek Lokasi Wisata(Studi Kasus Wisata Di Kota Palembang)", Vol. 3,No. 1, 2022.