JOURNAL OF DATA SCIENCE | Vol.2023:22
eISSN:2805-5160

Simulation and Visualization of TSP Using Ant Colony Optimization

Tri Basuki Kurniawan [1], Misinem [2], Astried[3], Joan Angelina Widians[4]

[1]Information Systems Study Program, Bina Darma University
[2]Computer Engineering Study Program, Bina Darma University
[3]Information Systems Study Program, University of Riau
[4]Informatics Engineering Study Program, Mulawarman University

***Email**: [1]tribasukikurniawan@binadarma.ac.id, [2]misinem@binadarma.ac.id, [3]astried@lecturer.unri.ac.id, [4]angel.unmul@gmail.com

Abstract

The Travelling Salesman Problem (TSP) is a well-known algorithmic problem its main objective is optimization. While naturally, it will require a significant amount of time and effort to finish a lot of complex work by hand such as TSP, one of the most exciting advances in software development is the discovery of optimization algorithms. When we are presented with a difficult assignment, optimization techniques can be used to ensure that the complex work is completed efficiently and quickly, including identifying the best solution to solve it. This study aims to simulate and visualize the TSP by developing a software simulation model. This model can be used to explain how to solve the traveling salesman problem by providing step-by-step instructions performed by users using ant colony optimization. With Windows as the operating system and the C# programming language in Visual Studio 2019, the program complies with the Extreme Programming (XP) software development method. The study concludes that optimization can be applied using programming language to provide users with comfortable and intelligible simulations and visualizations.

Keywords

Visualization, Travelling Salesman Problem, Ant Colony Optimization, Simulation Model

Introduction

The computer field is the most complicated and complex engineering field ever studied or discovered by humans and is also the fastest-growing field. It currently takes advantage of computers' magical abilities in almost all areas. Computers have colored and controlled our daily life activities. Whether it's a computer in the form of a personal computer (PC, personal computer), a laptop, or a smartphone, modern humans deal with computers and other intelligent devices almost daily. Even our lives depend on this equipment (Elawady et al., 2022). If we were to look again, we would see that the software that runs and manages the equipment is quite clever, not the hardware itself. The equipment that assists us in our daily jobs is managed and equipped with capabilities using software, which is created by an individual or team of programmers utilizing a particular programming language.

Submission: 29 November 2023; **Acceptance:** 4 December 2023

The expectations for hardware and software capabilities have grown dramatically in tandem with the diversity and complexity of today's modern human demands. In the past, we could get by with equipment that had basic functions (Okoye et al., 2023). However, as the workload and activities expand, it is also anticipated that the equipment's capability will improve. Consequently, there is a demand for and a requirement for equipment and software with good capabilities.

The development of optimization algorithms is one of the more interesting areas in software engineering. Many complex activities are either too difficult to accomplish by hand or would require a great deal of time and effort to accomplish by hand. Furthermore, one is frequently not certain to obtain the best or greatest option (Mikalef & Gupta, 2021). These difficult and complex tasks can be completed more quickly and efficiently due to the optimization algorithm. It even offers theoretical assurances for obtaining the optimal outcome. Almost all scientific disciplines, including engineering, physics, social sciences, economics, and business, can use this optimization procedure. Numerous issues in science, engineering, and economics can be formulated as optimization problems, such as maximizing revenues and quality or minimizing expenses, time, and hazards (Anis et al., 2007).

The traveling salesman problem is one of the issues that can be resolved by applying the optimization technique (TSP). In the field of optimization, TSP is a well-known issue (benchmark problem). The problem with TSP is that the seller needs to travel to multiple places, each with known travel times (Ajayi et al., 2022). All current cities must be viewed by the seller; each visit is limited to one. The issue is in the seller's ability to plan his itinerary such that the total distance traveled is the best minimum distance—that is, the shortest distance the seller will travel.

This research discusses a simulation model with visualization to solve TSP problems using an artificial intelligence approach, the ant colony optimization method, or Ant Colony Optimization (ACO) (Kurniawan, 2009).

Methodology

Software Development Methods

The software development method used in this study is the Extreme Programming (XP) method, one of the Agile methods. The basis of the XP method is writing program code and testing (Alshayeb & Li, 2006). Software developers widely use XP because it focuses more on user satisfaction than completing complete and thorough documentation that may not be needed during the early stages of software development, considering that documentation will continuously develop and change. XP allows customers to change their requirements even at the final stages of software development.

The main tasks performed in XP are (Fojtik, 2011):

- Planning and Organizing
- Design
- Compile program code
- Testing

Travelling Salesman Problem (TSP)

Mathematical problems related to TSP began to emerge around the 1800s. This problem was raised by two mathematicians, namely Sir William Rowan Hamilton from Ireland and Thomas Pennington Kirkman from England (Khan & Maiti, 2019). The general form of the TSP problem was first studied by mathematicians starting in the 1930s by Karl Menger at Vienna and Harvard (Gede et al., 2017). Hassler Whitney and Merril Flood at Princeton later developed this problem.

TSP, in general, is a complex optimization problem; whereas the variables increase linearly, the time required will increase exponentially. Following the path from one city to another, the TSP can be divided into two types. The symmetric TSP, if the way from city A to city B is the same as the distance from city B to city A. Meanwhile, the asymmetric TSP, otherwise. For asymmetric, the number of possible trip combinations (solutions) is $(n - 1)!$ Meanwhile, the number of possible explanations for symmetric ones is half that of asymmetric solutions.

The number of solutions for n cities in the symmetric TSP case is $(n - 1)!/2$. If we had 10 (n) cities, there would be 181,440 possible solutions; for 15 cities, there would be 43,589,145,600 (43 billion more) possible solutions. We can't examine and calculate each solution and choose the best one. Besides taking a very long time, it also requires much effort. It means that it is not efficient and effective if the search for the best solution is done manually. For this reason, an optimization algorithm is needed. This problem is also called the nondeterministic polynomial-time hardness problem (Fortnow, 1997) or the np-hard problem.

Process Optimization

Optimization is a process to achieve ideal or optimal results (practical value that can be achieved). In mathematics, optimization studies problems that try to find an actual function's minimum or maximum value. A variable value is systematically selected to achieve the optimal value, either the minimum or the maximum, providing the optimal (best) solution (Anis et al., 2007).

The concept of optimization has been used since prehistoric times. It can be proven by the presence of water channels found at prehistoric sites. These waterways are used to optimize water use. It indicates that optimization has been a part of human life for a long time. The problem of water regulation is still found today. It's just that the solution has used modern optimization methods.

Optimization is a search method that aims to find a solution to an optimization problem, which gives the optimal value that may depend on several constraints. Even though the definition of optimization is straightforward, very complex things must be resolved. For example, the solution to be achieved may consist of several different types of data, with stringent boundaries and a challenging search space (convoluted) with many potential solutions, as well as problem characteristics that may change over time or certain conditions (dynamic problems) or the quality of the objective being searched for optimal value occurs where there is a conflict between one objective and another (multi-objective problem).

Ant Colony Optimization (ACO)
Observations about the habits and behavior of ants have inspired many ant-based algorithms, which are used to solve combinatorial optimization problems in discrete search spaces. Apart from ants, many simple behaviors of other living things have inspired researchers to find new methods and algorithms in optimization and other fields of artificial intelligence (AI).

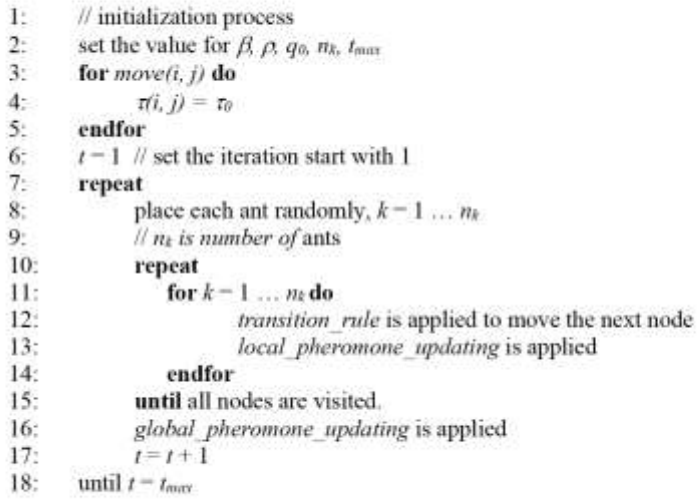
How can a group of ants find the shortest path between the food source they see and the nest where they live without any clues? Research on the habits and behavior of several ants shows a random search for food sources. However, after a food source is found, the activity of the ants becomes more organized, with more and more ants following the shortest trail. Gradually, eventually, all the ants follow the same path. There is an activity of 'transmitting knowledge' about the shortest distance from one ant to another.

Its 'transmitting knowledge' activity differs from one type of ant to another through direct contact or indirect communication. Most types of ants communicate through pheromone trails. Indirect communication is where ants change their environment (by placing a pheromone) to influence the behavior of other ants.

Ant Colony Optimization (ACO), which we will briefly refer to as ACO, is one of many algorithms inspired by the behavior of living things. Like the Genetic Algorithm, which takes the principles of genetic theory, Particle Swarm Optimization takes the behavior of a flock of animals, such as bees, which swarms on something and other algorithms.

The first algorithm designed by Dorigo in his Ph.D. dissertation is the Ant System (AS), which is the forerunner of other algorithms based on ant behavior. Furthermore, Dorigo made several additions and improvements to the AS algorithm by proposing the Ant Colony Optimization (ACO) Algorithm. These additions and enhancements include the three main processes in ACO, namely the state transition rule, global updating rule, and local pheromone updating rule, in the form of a mechanism for selecting tracks for ants, a reward process that will benefit the best solution (in this case, pheromone concentration) and a means for reducing pheromone concentrations by evaporation. For further information regarding the formulation for each rule used, see the research conducted by Kurniawan et al. (Sahni, 2008).

In simple terms, the ACO algorithm can be explained as in the pseudo-code in Figure 1 below.



```
1:   // initialization process
2:   set the value for $\beta$, $\rho$, $q_0$, $n_k$, $t_{max}$
3:   for $move(i, j)$ do
4:       $\tau(i, j) = \tau_0$
5:   endfor
6:   $t = 1$ // set the iteration start with 1
7:   repeat
8:       place each ant randomly, $k = 1 \dots n_k$
9:       // $n_k$ is number of ants
10:      repeat
11:          for $k = 1 \dots n_k$ do
12:              transition_rule is applied to move the next node
13:              local_pheromone_updating is applied
14:          endfor
15:      until all nodes are visited.
16:      global_pheromone_updating is applied
17:      $t = t + 1$
18:  until $t = t_{max}$
```

Figure 1. ACO Algorithm Pseudo-code

In Figure 1, the pseudo-code of ACO algorithms is shown. In step 2, the variables β, ρ, q_0, n_k, t_{max} are set with values shown in Figure 6. Next, for each pheromone matrix values are set with τ_0 where that value is the default value for the pheromone when the algorithm is started. Usually, we use the 1 divided by the total distance for one tour. In line 6, we start the iteration process. From line 1 until line 18, the process is repeated a maximum number of iterations we set in t_{max}. We start by placing ants at the start position, randomly. Next, we apply *transition_rule* and *local_pheromone_updating* for each ant, until all nodes are visited. Lastly, we apply *global_pheromone_updating* only for the best solution. At the end of iterations, we select the solution which has a short distance as the best solution.

Simulation Model Design

In the early stages of making a program, we first must design our program's interface (interface) or appearance. It will be more practical because we will build a program with a graphical display on Windows.

The appearance of our program is shown in Figure 2 below.

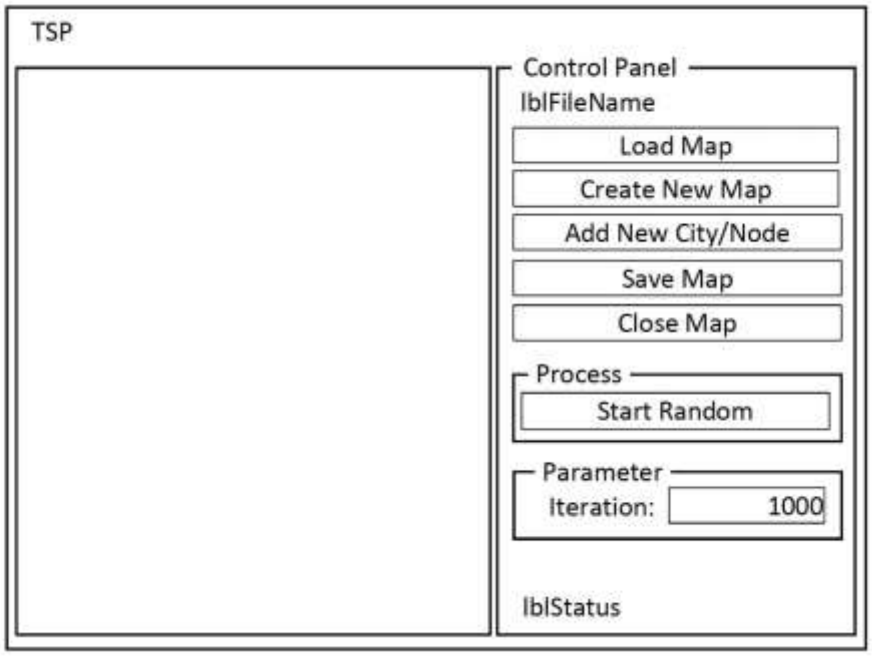
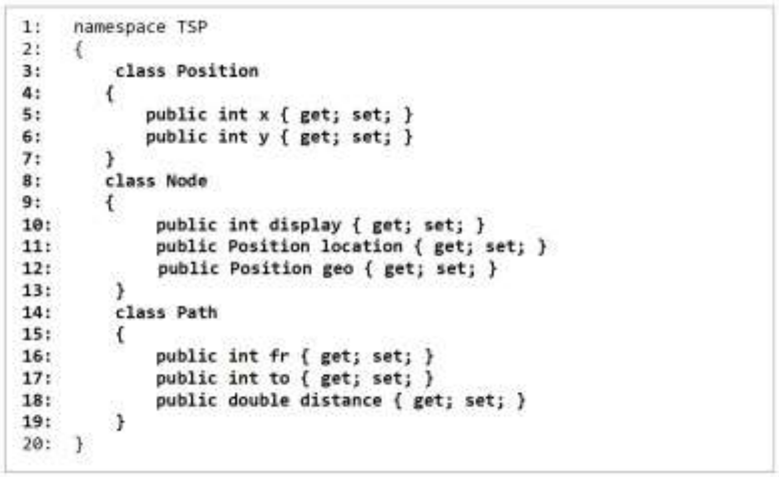


Figure 2. Main view mock-up design.

Next, the object designs used in this program can be seen in Figure 3 as Position, Node, and Path objects.



```
namespace TSP
{
    class Position
   {
        public int x { get; set; }
        public int y { get; set; }
   }
   class Node
   {
         public int display { get; set; }
         public Position location { get; set; }
         public Position geo { get; set; }
    }
    class Path
    {
         public int fr { get; set; }
         public int to { get; set; }
         public double distance { get; set; }
    }
}
```

Figure 3. Position, Node and Path object designs

As a media for the simulation, a display is created to display a simulation of the nodes that describe the cities that the seller will visit. The design of the simulation media used can be seen in Figure 4 and Figure 5 below.

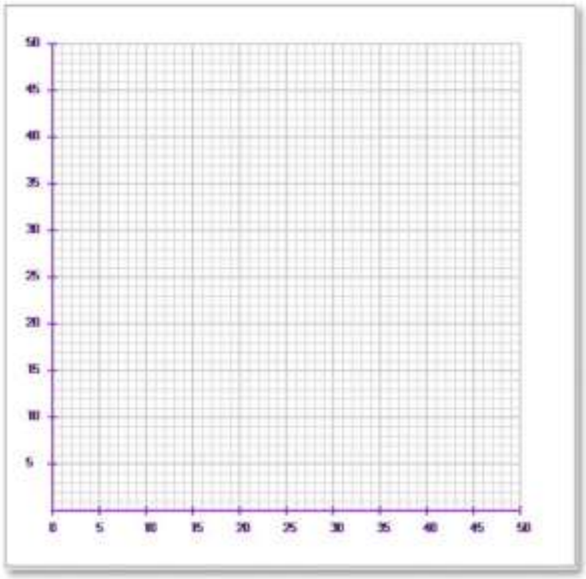
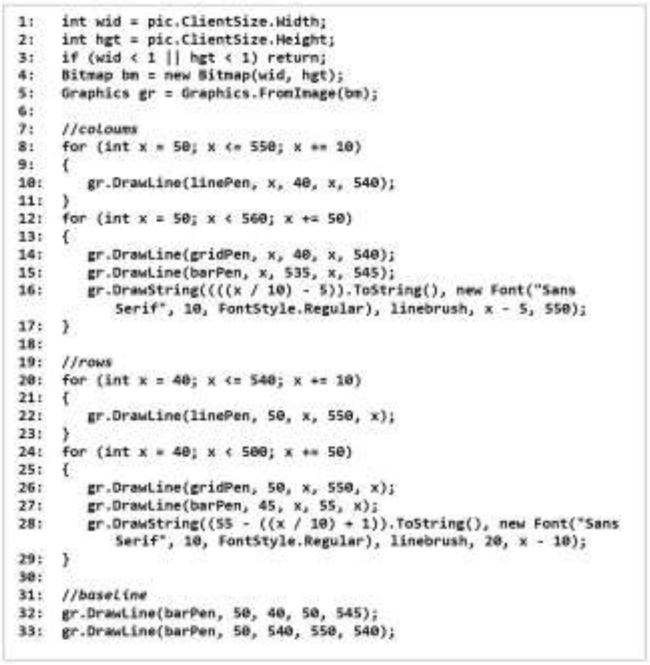


Figure 4. Simulation media design



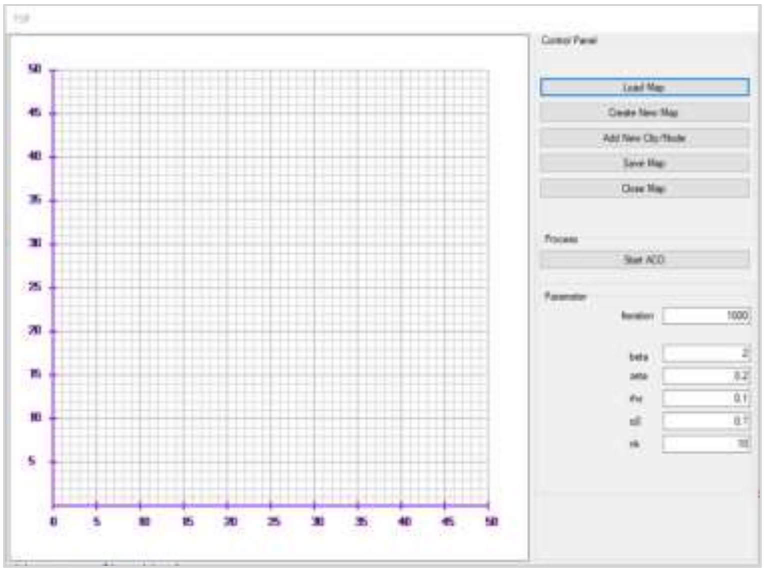
```
int wid = pic.ClientSize.Width;
int hgt = pic.ClientSize.Height;
if (wid < 1 || hgt < 1) return;
Bitmap bm = new Bitmap(wid, hgt);
Graphics gr = Graphics.FromImage(bm);

//coloums
for (int x = 50; x <= 550; x += 10)
{
   gr.DrawLine(linePen, x, 40, x, 540);
}
for (int x = 50; x < 560; x += 50)
{
   gr.DrawLine(gridPen, x, 40, x, 540);
   gr.DrawLine(barPen, x, 535, x, 545);
   gr.DrawString((((x / 10) - 5)).ToString(), new Font("Sans
      Serif", 10, FontStyle.Regular), linebrush, x - 5, 550);
}

//rows
for (int x = 40; x <= 540; x += 10)
{
   gr.DrawLine(linePen, 50, x, 550, x);
}
for (int x = 40; x < 500; x += 50)
{
   gr.DrawLine(gridPen, 50, x, 550, x);
   gr.DrawLine(barPen, 45, x, 55, x);
   gr.DrawString((55 - ((x / 10) + 1)).ToString(), new Font("Sans
      Serif", 10, FontStyle.Regular), linebrush, 20, x - 10);
}

//baseline
gr.DrawLine(barPen, 50, 40, 50, 545);
gr.DrawLine(barPen, 50, 540, 550, 540);
```

Figure 5. Code for creating simulation media.

Results and Discussion

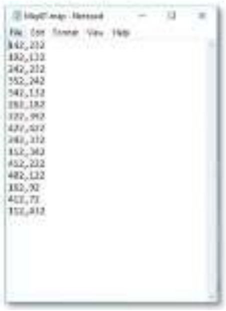
Using the Extreme Programming (XP) software development methodology and the C# programming language in Visual Studio Community 2019, our simulation and visualization were implemented in the previous concept. The platform consists of an Intel i-7 PC running Windows 10 with 16GB of RAM.

Figure 6 displays the simulation and visualization results after it is built. We also conducted testing material using simple data from several nodes with different locations. Figures 7 and 8 depict the simulation of the cities to be visited once the nodes are displayed in the program using the [Load Map] button.



Iteration 1000
beta 2
zeta 0.2
rho 0.1
q0 0.1
nk 10

Figure 6. Main view of the simulation program



```
142,212
182,132
242,212
352,242
542,132
282,182
322,992
427,427
262,312
152,362
452,212
482,122
182,92
412,72
152,412
```

Figure 7. List of node or city positions as problem trials.

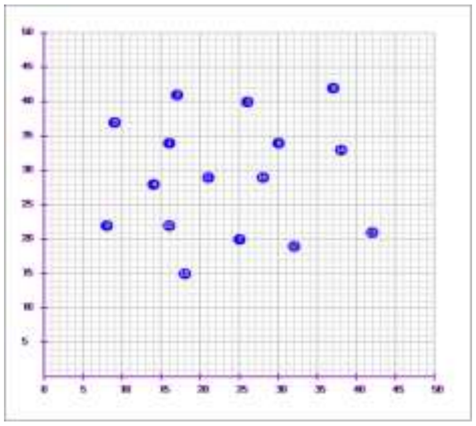
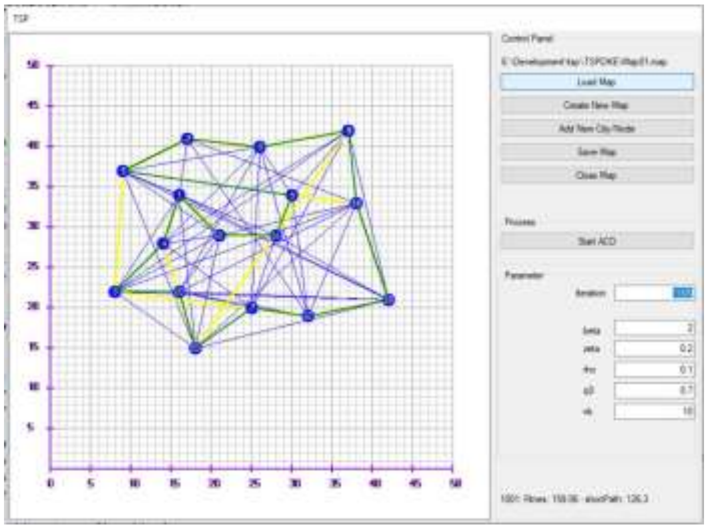


Figure 8. Simulation results of node/city positions in the simulation program

The results of the simulation using the ACO method can be seen in Figure 9 below. In Figure 9, you can see several paths with different colors, namely blue, showing the trail for each ant (in this study, using ten (10) ants, variable *nk*). Yellow is the best trail for each iteration (this research uses 1000 iterations, variable iteration). Lastly, green is the best trace line obtained in this simulation. Table 1 shows the best trace lines obtained.



beta 2
zeta 0.2
rho 0.1
q0 0.7
nk 10

1001: Rbws: 159.06 - shortPath: 126.3

Figure 9. Simulation results with simple trial data

Based on the collected data, it can be inferred that there are variations between the best trace length obtained and the best result trace length from the previous iteration, which was 32.76.

Table 1. Results of simulation solutions on simple test data.

Type	Trace line	Trace length
Green	2-3-8-14-13-12-7-15-10-9-4-1-11-16-6-5-2	126.30
Yellow	8-10-9-7-4-11-16-14-15-6-3-1-2-5-12-13-8	159.06

Conclusion

The results of our study show that the traveling salesman problem (TSP) can be effectively solved using simulation and visualization by employing the ant colony optimization (ACO) approach, producing favorable visual results. The program's simulation of finding the shortest path solution can provide the user with a clear image of the process by providing the process outcomes in each iteration to the simulation media. Every change in the results can be directly seen and verified by the user. If users could see which trail lines were most attractive to ants—those with higher values—it would be much more intriguing if the pheromone matrix were colored differently for each distinct range of values. For development purposes, a pheromone matrix can also be shown to illustrate how the value changes with each iteration.

References

Ajayi, B. A., Magaji, M. A., Musa, S., Olanrewaju, R. F., & Salihu, A. A. (2022). A Comparative Analysis of Optimization Heuristics Algorithms as Optimal Solution for Travelling Salesman Problem. *2022 5th Information Technology for Education and Development (ITED)*, 1–8. https://doi.org/10.1109/ITED56637.2022.10051627

Alshayeb, M., & Li, W. (2006). An empirical study of relationships among extreme programming engineering activities. *Information and Software Technology*, *48*(11), 1068–1072. https://doi.org/10.1016/j.infsof.2006.01.005

Anis, M., Nandiroh, S., & Utami, A. (2007). OPTIMASI PERENCANAAN PRODUKSI DENGAN METODE GOAL PROGRAMMING. *Jurnal Ilmiah Teknik Industri*, *5*.

Elawady, M., Sarhan, A., & Alshewimy, M. A. M. (2022). Toward a mixed reality domain model for time-Sensitive applications using IoE infrastructure and edge computing (MRIoEF). *Journal of Supercomputing*, *78*(8), 10656–10689. https://doi.org/10.1007/s11227-022-04307-8

Fojtik, R. (2011). Extreme programming in development of specific software. *Procedia Computer Science*, *3*, 1464–1468. https://doi.org/10.1016/j.procs.2011.01.032

Gede, L., Candrawati, A., Gede, G. A., & Kadyanan, A. (2017). OPTIMASI TRAVELING SALESMAN PROBLEM (TSP) UNTUK RUTE PAKET WISATA DI BALI DENGAN ALGORITMA GENETIKA. In *Jurnal Ilmiah ILMU KOMPUTER Universitas Udayana: Vol. X* (Issue 1).

Khan, I., & Maiti, M. K. (2019). A swap sequence based Artificial Bee Colony algorithm for Traveling Salesman Problem. *Swarm and Evolutionary Computation*, *44*, 428–438. https://doi.org/10.1016/j.swevo.2018.05.006

Mikalef, P., & Gupta, M. (2021). Artificial intelligence capability: Conceptualization, measurement calibration, and empirical study on its impact on organizational creativity and firm performance. *Information & Management*, *58*(3), 103434. https://doi.org/https://doi.org/10.1016/j.im.2021.103434

Okoye, K., Hussein, H., Arrona-Palacios, A., Quintero, H. N., Ortega, L. O. P., Sanchez, A. L., Ortiz, E. A., Escamilla, J., & Hosseini, S. (2023). Impact of digital technologies upon teaching and learning in higher education in Latin America: an outlook on the reach, barriers, and bottlenecks. *Education and Information Technologies*, *28*(2), 2291–2360. https://doi.org/10.1007/s10639-022-11214-1

Sahni, Sartaj. (2008). *Proceedings of the fourth IASTED International Conference on Advances in Computer Science and Technology, April 2-4, 2008, Langkawi, Malaysia*. International Association of Science and Technology for Development.