International Journal of Multidisciplinary Research and Growth Evaluation.

Application of Discrete Event Simulation to Enhance the Efficiency of the Ant Colony Optimization Algorithm

Hoang Van Bay [1], **Le Ngoc Giang** [2*], **Nguyen Van Thong** [3]
[1-3] Faculty of Fundamental Technical, AD-AF Academy of Vietnam, Vietnam

* Corresponding Author: **Le Ngoc Giang**

Article Info

ISSN (online): 2582-7138
Volume: 06
Issue: 01
January-February 2025
Received: 17-12-2024
Accepted: 06-01-2025
Page No: 1160-1165

Abstract
The application of the Ant Colony Optimization (ACO) algorithm to solve the the Traveling Salesman Problem (TSP) has been extensively studied by scientists worldwide. However, implementing the algorithm faces challenges due to the randomness in the departure and movement times of ants, as well as the limitations of computer hardware. These factors reduce the algorithm's convergence ability and overall effectiveness. This paper proposes an approach to implement the algorithm by utilizing discrete event simulation (DES). Artificial ants are modeled to depart and move completely randomly, closely mimicking the behavior of natural ants. This approach accelerates the algorithm's convergence, minimizes the likelihood of falling into local optima, and enhances overall performance. The simulation results clearly demonstrate the advantages of this method.

DOI: https://doi.org/10.54660/.IJMRGE.2023.4.5.1160-1165

Keywords: Ant Colony Optimization, Traveling Salesman Problem, Discrete Event Simulation, Algorithm Convergence

Introduction

In graph theory, the shortest path problem is the problem of finding a path between two vertices such that the sum of the weights of the edges forming the path is minimized. It can be described as follows: Given a weighted graph with a set of vertices denoted as $V=(v_1,v_2,...,v_n)$ and a set of edges denoted as E, with a weight function $f:E\rightarrow R$, we need to find a path $P \subseteq EP$ from the vertex vs to the destination vertex vd in V, such that the sum $\sum_{eij\in P} f(eij)$ is minimized, with the condition that each vertex in P is visited exactly once. A typical example of the shortest path problem is the Travelling Salesman Problem, which is an NP-Hard problem in the class of discrete optimization problems. The problem is stated as follows: A salesperson needs to deliver goods to nnn cities, starting from any city and then visiting other cities to make the deliveries and return to the starting point. Each city must be visited exactly once, and the distances between cities are given. The task is to find the shortest route to help the salesperson.

The TSP was first introduced in 1930 and is one of the most optimized problems with numerous practical applications, such as in planning, logistics, microchip production, and gene analysis in biology [14, 15, 16]. In these applications, the concept of a city can be replaced by customers, soldering points on a circuit board, or DNA fragments in genes. The concept of distance can be represented by travel time or cost, or by the comparison between DNA fragments.

To solve the TSP, exact methods can be used when the number of cities is small [19]. However, when the number of cities is large, exact methods are no longer suitable due to the excessive time required for solving, which is unacceptable in practice. Typically, the approach to solving large TSP problems is to use approximation methods, which provide results with an acceptable margin of error and significantly faster solution times compared to exact methods.

For approximate solutions to the TSP, nature-inspired algorithms can be employed, among which ACO stands out. Several approaches have been discussed in [17] and [18], where the authors have proposed various methods for parameter selection and scent trail updating. However, during the algorithm's implementation, the updating of the pheromone concentration does not accurately reflect reality, as the artificial ants do not fully resemble real ants due to computational limitations, such as the limited number of threads on a computer. Therefore, an approach is needed to ensure that artificial ants move in a manner closer to real ants.

A DES algorithm is proposed to address this limitation. By applying DES, the randomness of the artificial ant colony is increased, minimizing the risk of falling into local optima and accelerating convergence. The results are validated through simulations.
The remainder of the paper is structured as follows: Section 2 provides a brief introduction to the ACO algorithm for solving the TSP. Section 3 presents the development of the ACO algorithm with DES application. Section 4 outlines the simulation program and presents the achieved results. Conclusions and suggestions for future research are provided in Section 5.

1. Ant colony optimization algorithm

ACO is a nature-inspired optimization method based on simulating the behavior of ant colonies in nature to solve complex optimization problems. Ants are always able to find the shortest path from their nest to a food source. The means of communication to find the most efficient path is their pheromone trail. Over time, this pheromone trail gradually disappears, but it can receive reinforcement if other ants persist in following the same path.
To mimic the behavior of real ants, we construct artificial ants that also produce pheromone trails on their paths and have the ability to follow these trails by choosing paths with higher pheromone concentrations. Based on this characteristic, developed the Ant System (AS). Subsequent versions of ACO have been developed, including the Ant Colony System (ACS) and Max-Min, by altering the methods for updating and evaporating the pheromone concentrations [2, 6, 7].
Initially, the pheromone concentration on each edge (i, j) is initialized with a constant value c, or it can be determined by the following formula:

$$\tau_{ij} = \tau_o = \frac{m}{C^{nn}}, \forall(i,j)$$

Where:
τ_{ij}: Pheromone concentration on edge (i,j)
m: Number of ants
C^{nn}: Length of the path given by the nearest neighbor search method
At vertex i, an ant k will choose a vertex j, which has not been visited yet, from the neighborhood set of i according to a probability distribution rule, determined by the following formula:

$$p_{ij}^k = \frac{[\tau_{ij}]^\alpha [\eta_{ij}]^\beta}{\sum_{l\in N_i^k} [\tau_{il}]^\alpha [\eta_{il}]^\beta}, j \in N_i^k$$

Where:
p_{ij}^k: Probability that ant k chooses edge (i,j)
α: adjustment factor the influence of τ_{ij};
β: adjustment factor the influence of η_{ij};
$\eta_{ij} = \frac{1}{d_{ij}}$: Heuristic information;
d_{ij}: Distance between vertices i and j;
N_i^k: Set of unvisited neighboring vertices of i by ant k

During the search process, the pheromone trails on each edge are updated, as they are modified due to evaporation and accumulation of pheromone when ants travel along that edge. After each iteration, the pheromone trail on each edge is updated according to the following formula:

$$\tau_{ij}(t+1) = (1-\rho) \times \tau_{ij}(t) + \sum_{k=1}^{m} \Delta\tau_{ij}^k(t), \forall(i,j)$$

Where:
$0 < \rho \leq 1$: evaporation rate of the pheromone trail;
$\Delta\tau_{ij}^k(t)$: amount of pheromone left by ant k on edge (i,j)

$$\Delta\tau_{ij}^k = \begin{cases} \frac{Q}{f(k)} & \text{If the ants do not pass} \\ 0 & \text{If the ants pass} \end{cases}$$

Where:
Q: constant
$f(k)$: length of the path constructed by ant k
When applying the ACO method to each specific problem, there are three factors that determine the effectiveness of the algorithm: constructing an appropriate structural graph, selecting heuristic information, and choosing the pheromone update rule.

2. Application of des algorithm

2.1. Description of des

The system model is divided into continuous models and discrete event models. In continuous system simulations, the time step is fixed at the start of the simulation. Time increases in equal increments, and the values change directly over time. In this type of simulation, the values that reflect the system's state are simulated at any given time, and the simulation time increases uniformly from one step to the next, as shown in Figure 1.

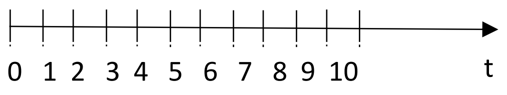


Fig 1: Fixed time step in continuous simulation In discrete event simulations, the system changes its state only when events occur and only when those events take place. Time passing does not directly affect the simulation. Unlike a continuous simulation, the simulation time progresses from one event to the next, and the time between events is not guaranteed to be equal, as in Figure 2.

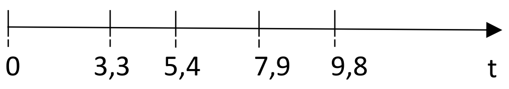


Fig 2: Time step in discrete event simulation

DES simulates the activities of a system as a sequence of discrete events over time, where each event occurs at a specific moment and marks a change in the system's state [12, 20, 21]. Discrete event simulation consists of the following steps:
Step 1: Initial setup. Initialize the system's state variables, initialize the clock, and schedule the first event.
Step 2: Repeat. Set the clock for the next event, process the next event, remove it from the event list, and update the statistics.
Step 3: Generate statistical reports.
Next, we will examine the application of discrete event simulation to artificial ant colonies.

2.2. Application of DES for the Artificial Ant colony model

For simplicity, we consider the case of an ant colony consisting of five ants searching for food, where the movements of the ants are entirely random. This problem is then generalized for a colony of n ants. The five ants move independently as described in Figure 3.

Let N be the number of vertices that the ants need to traverse, t_{ij} be the time it takes for the ith ant to pass through the jth vertex. Initially, the ants all start from their nest (vertex 0) and move independently. For simplicity, we consider the starting point to be ant 1 and the final one being ant n (n=5). Over time, the event of an ant moving is continuously triggered, ensuring that all ants will traverse the N vertices.

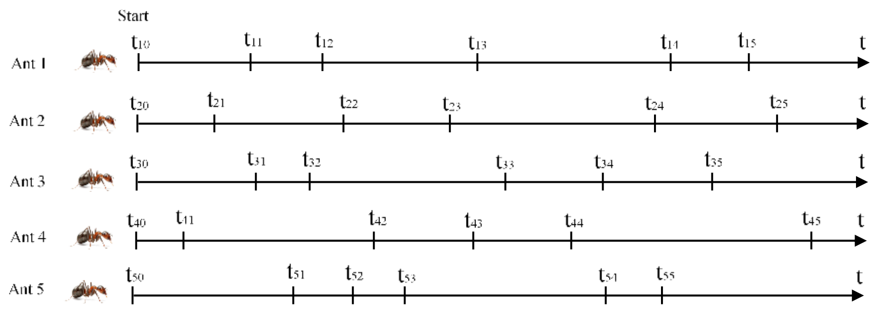

Start
Ant 1
t10 t11 t12 t13 t14 t15 t
Ant 2
t20 t21 t22 t23 t24 t25 t
Ant 3
t30 t31 t32 t33 t34 t35 t
Ant 4
t40 t11 t42 t43 t44 t45 t
Ant 5
t50 t51 t52 t53 t54 t55 t

Fig 3: The random movement process of artificial ants according to event-driven simulation

3. Simulation and results achieved

3.1. Building algorithm flowchart

The DES algorithm is implemented as shown in Figure 4. We see that the DES algorithm allows the ants to depart at random times and at any vertex, updating the local pheromone exactly when the ants pass through the vertices and the global one when the ants complete the loop. That will increase the randomness, minimizing the possibility of falling into local extrema.

The algorithm simulates the Ant Colony Optimization process using a discrete event simulation framework. The detailed steps are as follows:

- **Step 1:** Initialization

Set up the initial parameters, including the ant colony, pheromone levels, and environmental parameters. Schedule the first event.

- **Step 2:** Retrieve the Event

Extract the next event from the event queue for processing.

- **Step 3:** Check if All Ants Have Visited All Nodes

If True: Update pheromone levels based on the paths traveled and proceed to check for remaining events.

If False: Proceed to the next vertex selection for the ants.

- **Step 4:** Select the Next Vertex

Use the ACO heuristic to determine the next vertex for the ant's movement.

- **Step 5:** Evaluate Distance

If the distance is greater than the threshold, continue processing and consider this path.

If less than or equal to the threshold: Remove the selected ant from further simulation.

- **Step 6:** Update Pheromones

Update the pheromone levels on edges based on the quality of paths traveled by all ants.

- **Step 7:** Add New Event

Schedule additional events representing subsequent movements or actions of the ants.

- **Step 8:** Check for Remaining Events

If no events remain: Perform a final pheromone update and check if the iteration has ended.

If events remain: Return to Step 2 to continue processing.

- **Step 9:** Iteration Completion Check

If the iteration is complete: Move to the shortest path determination step.

If not complete: Return to Step 2 to process further events.

- **Step 10:** Identify the Shortest Path

After completing all iterations, compute the shortest path based on accumulated pheromone levels.

- **Step 11:** Termination

End the simulation and save the results.

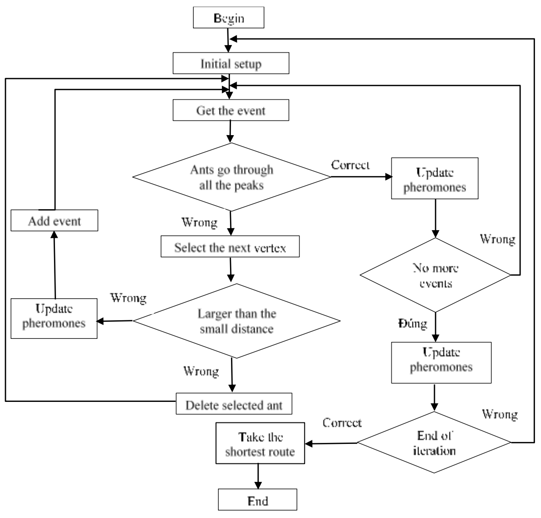
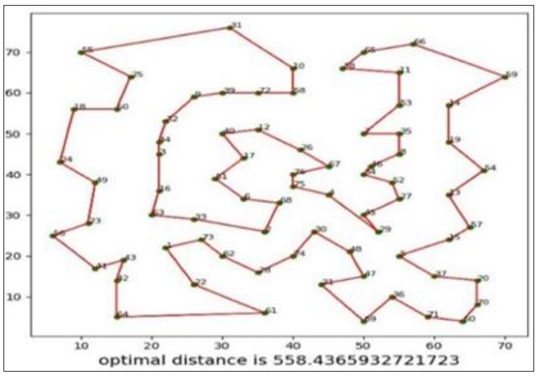


Fig 4: Flowchart of ACO implementation using DES algorithm

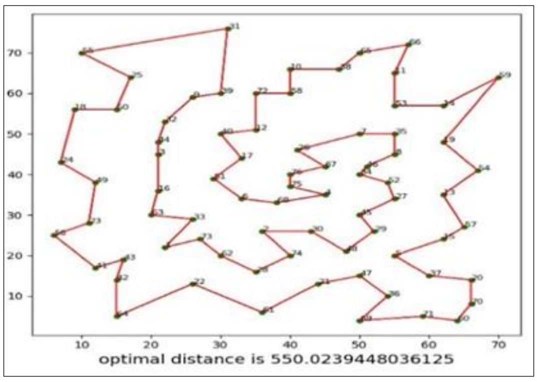
Table 1: Comparison of simulation results with eil76 and Berlin52 standard library

	Name	Best	Achieved	Error Rate (%)	Time (s)
1nd time	Eil76	538	558.4	3.7	33.2
	Berlin52	7542	7658.9	1.54	42.0
2nd time	Eil76	538	550.0	2.23	34.7
	Berlin52	7542	7657.8	1.53	43.5
3nd time	Eil76	538	556.6	3.4	37
	Berlin52	7542	7713.0	2.26	43.0
4nd time	Eil76	538	565.2	5.0	35.6
	Berlin52	7542	7716.8	2.3	46.3

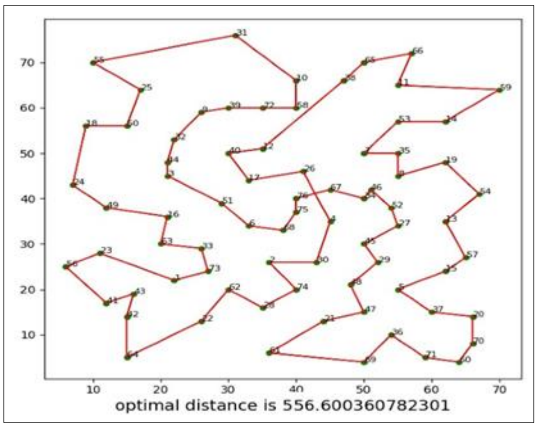
3.2. Results



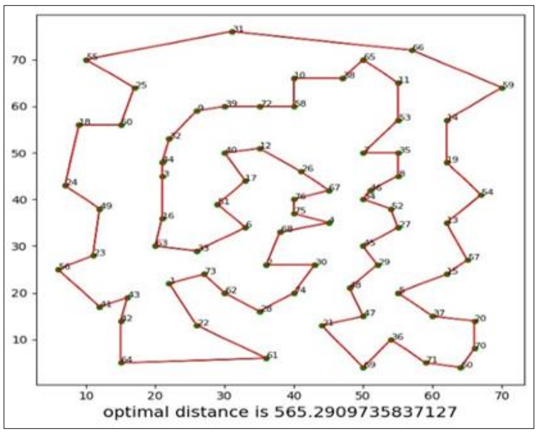
a) 1nd time



b) 2nd time

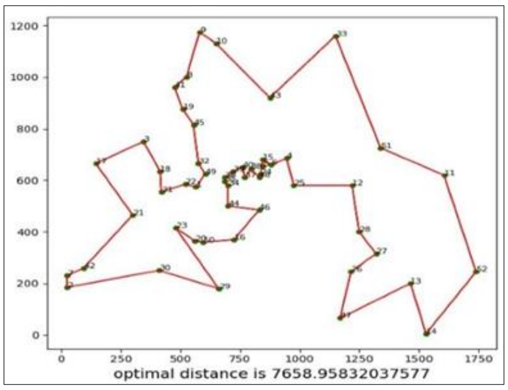


c) 3nd time

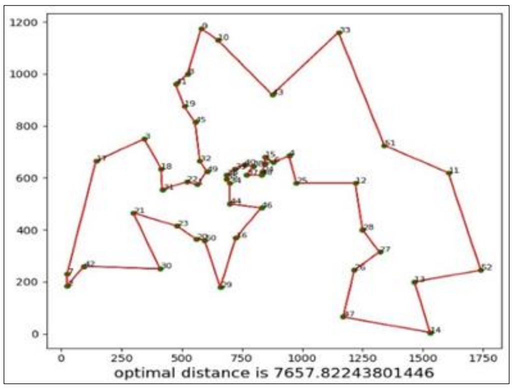


d) 4nd time

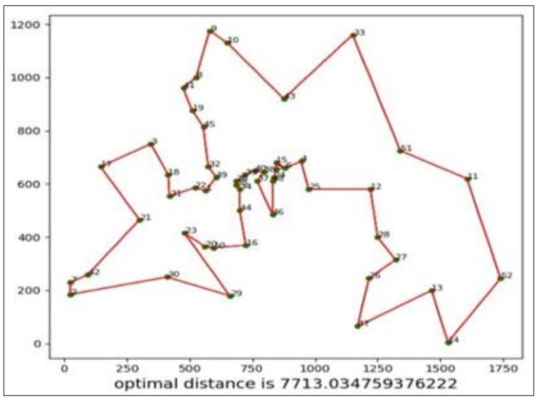
Figure 5: Pathfinding results for the Eil76 dataset



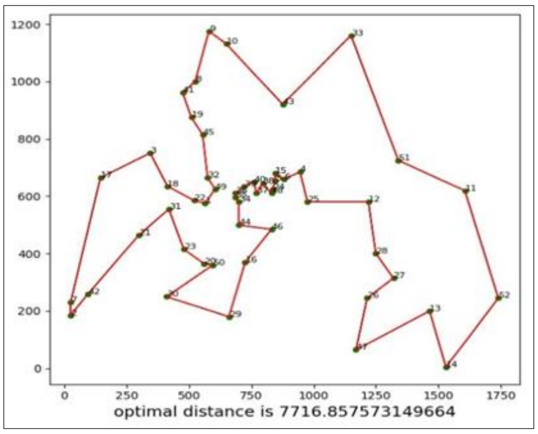
a) 1nd time



b) 2nd time



c) 3nd time



d) 4nd time

Fig 6: Pathfinding results for the Berlin52 dataset

To implement the DES algorithm, several simulation software tools such as Arena, MATLAB/Simulink, and others can be used. In this study, the author utilized the Python programming language (source code can be downloaded at) [22] with data from the TSPLib standard library (a library of sample data for the TSP problem), which includes city coordinates and pre-calculated optimal values. In this paper, two sample coordinate sets, Eil76 and Berlin52, were used.

Table 1 shows the simulation results on a computer with 20 ants and 200 iterations, using the Eil76 and Berlin52 coordinate sets. The results show that after 4 trials, for the Eil76 coordinate set, the lowest error was 2.23, and the highest was 5.0. For the Berlin52 coordinate set, the lowest and highest errors were 1.53 and 2.3, respectively. These errors are acceptable in practical conditions, while convergence time is fast.

The results of the paths are graphically represented with the Eil76 and Berlin52 coordinate sets in Figures 5 and 6, respectively.

4. Conclusion

This paper proposes a method utilizing discrete event simulation to enhance the effectiveness of the Ant Colony Optimization algorithm for the Traveling Salesman Problem. By combining this approach with appropriate parameter selection techniques, promising results were achieved. Compared to optimal results published in the TSPLib benchmark library, the proposed algorithm demonstrated high accuracy, with errors below 5% and faster convergence times.

However, these results represent only an initial stage of the research. In the future, the authors plan to further develop the method by integrating automatic parameter selection, adaptive adjustment rules, and natural evolution mechanisms. While the proposed algorithm is applied specifically to the TSP in this study, it can be extended for use with ACO in solving other real-world optimization problems.

5. References

1. Wang J, Li H, Chen H. Mixed ant colony algorithm for vehicle routing problem with time windows. ISSN:1662-8985. 2013 Jun 13.
2. Yigit T. Using the Ant Colony Algorithm for Real-Time Automatic Route of School Buses. The International Arab Journal of Information Technology. 2016 Sep 5.
3. Zhang S, Zhang Y. A Hybrid Genetic and Ant Colony Algorithm for Finding the Shortest Path in Dynamic Traffic Networks. ISSN 0146. 2017 Sep 29.
4. Chin T, Abbou F, Tat E. Simulation Study of a Heuristic Near-Maximum Ant-Based Dynamic Routing. The International Arab Journal of Information Technology. 2008;5(3):230-233.
5. Clarke G, Wright J. Scheduling of Vehicles from a Central Depot to a Number of Delivery Points. Operations Research. 1964;12(4):568-581.
6. Dorigo M, Gambardella L. Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem. IEEE Transactions on Evolutionary Computation. 1997;1(1):53-66.
7. Ekin E, Yakhno T. A Case Study of Adopting Ant System to Optimization Problems. The Tenth Turkish Symposium on Artificial Intelligence and Neural Networks, TAINN2001. Gazimagusa, T.R.N.C.; 2001. p. 20-25.
8. Van Breedam A. An Analysis of the Effect of Local Improvement Operators in Genetic Algorithms and Simulated Annealing for the Vehicle Routing Problem. RUCA Working Paper 96/14. University of Antwerp, Belgium; 1996.
9. Badeau P. A Tabu Search Heuristic for the Vehicle Routing Problem with Soft Time Windows. Transportation Science. 1997;31(2):170-186.
10. Rizzoli A, *et al*. Ant Colony Optimization for Real-World Vehicle Routing Problems. Swarm Intelligence. 2007;1(2):135-151.
11. Reimann M, *et al*. D-ants: Savings based Ants Divide and Conquer the Vehicle Routing Problem. Computers and Operations Research. 2004;31(4):563-591.
12. Fishman GS. Discrete-Event Simulation: Modeling, Programming, and Analysis.
13. Brucato C. The traveling salesman problem. Sonoma State University, 2010. M.S., University of Pittsburgh, 2013.
14. Chan D. IC insertion: an application of the travelling salesman problem. International Journal of Production Research. 1989 Oct;27(10):1837-1841.
15. Venkatesh D. Generation of Genetic Maps Using the Travelling Salesman Problem. International Journal of Scientific & Engineering Research. 2014 Jun;5(6).
16. Stuzle T. ACO Algorithms for the Traveling Salesman Problem. Universite Libre de Bruxelles, Belgium.
17. Brezina I. Solving Traveling Salesmanship Problem (TSP) Using Ant Colony Optimization. International Journal of Engineering and Technical Research. 2018

Jul.
18. Priestley HA, Ward MP. A Multipurpose Backtracking Algorithm. 2003 Jan.
19. Robinson S. Simulation - The practice of model development and use. Wiley; 2004.
20. Matloff N. Introduction to Discrete-Event Simulation and the SimPy Language. 2013 Jan. Available from:
21. https://github.com/vanbayhoang/DES_TSP.