International Journal of Engineering Trends and Technology
ISSN: 2231-5381/ https://doi.org/10.14445/22315381/IJETT-V74I11P122

Volume 74 Issue 11, 329-342, November 2026
© 2026 Seventh Sense Research Group®

Original Article

Performance Evaluation of Metaheuristic Algorithms on Electric Vehicle Route Optimization

Khalid Ashraf[1*], Binod K. Sharma[1], Shashank Pushkar[1]

[1]*Department of Computer Science and Engineering, BIT Mesra, Ranchi, Jharkhand, India*

**Corresponding Author: khalidashraf@bitmesra.ac.in*

Received: 11 October 2025 Revised: 16 September 2026 Accepted: 30 September 2026 Published: 25 November 2026

Abstract - *Electric Vehicles (EVs) are gaining global traction due to environmental concerns and energy needs. Electric Vehicles (EVs) are considered as a sustainable solution to mitigate environmental concerns these days. The objective of using Electric Vehicles (EVs) is to reduce fossil fuel dependency. So, navigating these vehicles efficiently needs a fundamental shift from traditional routing standards. Traditional routing algorithms like Dijkstra's, designed for fossil-fuel vehicles, may not identify energy-efficient EV paths due to regenerative braking and battery dynamics. This standard shortest-path framework clearly fails to identify truly energy-optimal paths for electric vehicles. A major shortcoming of these methods is that they fail to accurately represents complex battery behaviour. They do not sufficiently consider non-linear battery dynamics, varying State-of-Charge (SOC) conditions, or the energy recovered through regenerative breaking. This paper evaluates two meta-heuristic approaches called Ant Colony Optimization (i.e. ACO) and Particle Swarm Optimization (i.e. PSO) to address EV routing. This study presents a comprehensive mathematical formulation of Routing Problem for Electric Vehicles (EVRP). The study also presents a detailed comparative evaluation of both algorithms (ACO and PSO). Both algorithms are carefully tested against variable problem sizes ranging from 5 to 25 customer nodes and evaluated across extreme operational bounds from 100 to 10000 iterations. The performance of both methods is analysed comparatively across various criteria such as number of iterations, time taken in computation, best cost value and average solution cost. Experimental results show a clear divergence in performance. Specifically, PSO shows a profound superiority over ACO. These findings offer practical guidance for designing real-time, energy-efficient navigation system for electric vehicles.*

Keywords - *ACO, Metaheuristic algorithms, PSO, Vehicle Routing Problem (VRP), Energy-Efficient routing.*

1. Introduction

Drivers often have multiple route options when traveling from an origin station to the destination node on a known map. The most practical choice typically minimizes either Light Travel Time (LTT) or Light Travel Distance (LTD). In some cases, a balance between these two factors is sought, as optimizing both simultaneously may not be feasible. Traditionally, focusing on either distance or travel time indirectly leads to energy-efficient route selection, especially for gasoline or diesel vehicles, where longer travel times or distances result in increased fuel consumption. However, this principle does not apply to Electric Vehicles (EVs), which can regenerate energy under certain conditions. Electric vehicles utilize efficient electric-motor that can operate as generator during coasting, downhill travel, or braking, allowing them to recover kinetic or potential energy and store it in their high-voltage batteries. This regenerative braking [1] can enhance a vehicle's range by over 20%. Currently, EVs are seen as a leading solution for reducing carbon emissions. Approximately two-thirds of all oil used in the United States is used in transportation. A shift to electric vehicles could potentially cut total CO_2 emissions by nearly half [2]. In Europe, transportation accounts for 73% of oil consumption, with road transport contributing to 25% of CO_2 emissions [3].

In recent years, vehicle manufacturers have significantly invested in electric vehicles, driven by stricter regulations and environmental advocacy. Major manufacturers, including Tesla, Nissan, and Chevrolet, have announced plans for new electric-vehicle models for the coming years. This growing interest highlights the significance of studying energy-efficient routing for EVs. The unique constraints of EVs, such as their high-density lithium-ion batteries, introduce complexities in routing. These batteries have specific requirements regarding thermal conditions, charge-and-discharge velocities, and electrical constraints [4], which can vary with ambient conditions and the State-of-Charge (i.e. SOC). For instance, a battery (power cell) having a high state of charge can withstand high discharge rates, but its charge rates may be limited. All these dynamic limitations contribute to the non-linear nature of routing problems for EVs, prompting researchers and manufacturers to seek innovative

 This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/)

solutions. The subsequent sections will give details on why classic shortest-path techniques such as "Dijkstra's algorithm" may not be appropriate to address energy-efficient routing for electric transport. Numeric instances will illustrate that minimizing spatial distance does not always equate to the optimal energy-conservation route. A comprehensive literature review of study done so far and a specification of the problem will follow, along with an introduction to metaheuristic approaches, particularly PSO and ACO, for tackling such issue. The rest sections of this paper have been ordered as follows: Section 2 talks about the need for energy-efficient routing, Section 3 reviews previous research techniques to solve the EVRP, and Section 4 talks about both metaheuristic algorithms (ACO and PSO). Section 5 presents the problem definition and formulation. Experimental Results and Comparison Between ACO and PSO are discussed in Section 6. Finally, Section 7 will conclude with suggestions and propose future research directions.

2. The Need for Energy-Efficient Routing

Numerous commercially available routing techniques are accessible to consumers today, including popular services like Google Maps and MapQuest, which can be accessed online. Abouslaeiman et. al [5] have compared between the performance of time-pulsed charging framework against a conventional constant-current constant-voltage baseline. The primary benchmarking criteria focus on three distinct operational parameters such as charging times, energy capacity, and dynamic cell temperature changes. Many modern vehicles are equipped with Global Positioning System (GPS) receivers that store map data on a flash drive, typically providing the shortest distance routing options.

These GPS systems often rely on Dijkstra's algorithm or similar methods to determine the shortest path. Dutch computer scientist Edsger Dijkstra created the greedy search method known as Dijkstra's algorithm in 1959 [6], used to obtain the shortest path on graphs, making the same well-suited for routing applications. For Dijkstra's algorithm to function effectively, it requires all edge costs to be positive, except in scenarios where negative edges are solely connected to the source. If negative edge costs are present, the algorithm may fail to yield the near-optimum answer. Let us take an example of routing problem shown in Figure 1, which reveals this limitation. The diagram presented above represents a directed graph, showing the energy consumed by an electric vehicle as EV goes from the node available on left side to the right node. The numbers between the nodes designate the amount of energy consumed, while routes marked with negative values signify that the EV is descending, thereby generating energy. The primary purpose is to determine the fastest way from point "A" to the destination point "C." According to the graph, the quickest route is A → D → B → C. This route results in a net cost of -1. The execution of Dijkstra's algorithm on the routing network shown in Figure 1, following several core steps are taken into consideration:

1. A value of zero is assigned to the source point 'A' and ∞ (infinity) to the rest nodes (as shown in Figure 2(b).
2. Node 'A' is expanded and the cost values to all directly connected nodes are computed (illustrated in Figure 2(c).
3. Node 'C' is then expanded, which results in no changes to the cost values (as seen in Figure 2(d).
4. Node 'B' is subsequently expanded, which also yields no changes (shown in Figure 2(e).
5. Finally, node 'D' is expanded, which updates the routing cost of reaching node 'B' to -2 (As depicted in Figure 2(f).

Figure 2 shows how operational sequence of Dijkstra's algorithm moves across these stages. According to the results, Dijkstra's algorithm calculates that the most cost-effective way to move from point A to point B is -2. However, it incorrectly calculates the cost from 'A' to 'D' as +2 and the cost from 'A' to 'C' as 0. Both of these calculations are inaccurate. To determine the shortest path, the Bellman-Ford algorithm is particularly effective in situations where there are negative routes originating from a single source vertex [4, 7]. This algorithm can successfully identify negative nodes; however, if a negative (-ve) cycle can be reached from the source point, it is unable to determine the shortest path. Additionally, the Bellman-Ford algorithm generally operates more slowly than Dijkstra's algorithm. This has been mainly used in contexts where negative cycles are a concern, as it halts once such a cycle is detected. Although the Bellman-Ford algorithm could be adapted for energy-efficient routing in electric vehicles. If we do any sort of modifications over here, it might increase the complication of the algorithm. Incorporating the power cell constraints in EVs adds to the complexity of energy-efficient routing. A basic network routing example is presented in Figure 3(a), where the purpose is to navigate a vehicle from source node 'S' to termination node 'D'. Two potential routes are available: the first one is through intermediate node "a" and the second one is through node "b", these two alternatives have an equal cumulative cost of +1. If the EV starts at node 'S' with a fully charged battery (100% state of charge, SOC) and a battery-capacity of + 4, the path via vertex "a" is preferable, allowing the vehicle to reach node "D" with a +4-battery (power cell) capacity. However, if the EV takes the route via node 'b', it will reach point "D" with a battery-capacity of +2. If there is a case where the electric vehicle has a low state-of-charge (which is close to zero percent), the routing algorithm should opt for the route through node 'b' to avoid depleting the battery before arriving at the destination. Another scenario is illustrated in Figure 3(b). Here, an EV begins its journey from source node "S" and wants to reach destination node "D". The distances between these specific network nodes are indicated on the graph. A method focused solely on minimizing distance would select the route via node "a". Still, if vertex "D" is positioned at a Lower-Potential-Energy (LPE) level than point "S", this may not be the best option for an EV. If the path through node 'a' entails a severe ascent and a subsequent steep descent, but the path through node 'b' features a gradual decline, the energy-

efficient route for the EV would be through point 'b'. This is because a steep downhill section would necessitate friction braking, leading to a loss of recoverable potential energy. The factors outlined above drive researchers to investigate innovative strategies for tackling the energy-efficient routing challenge faced by Electric Vehicles (EVs). One particularly promising area that has garnered significant interest, especially with advancements in computational capabilities, is the use of metaheuristic algorithms. These algorithms are favoured for their versatility and adaptability across various problem domains. The following section will review related research, provide an overview of metaheuristic algorithms, and then concentrate especially on ACO and PSO as active solutions aimed at the routing problem in EVs.

3. Related Works

Research on energy-efficient routing specifically for Electric Vehicles (EVs) is still relatively limited. Several factors have hindered progress in this area, including restricted access to Global Positioning System (GPS) data, limited availability of detailed electronic maps, and major automakers have only

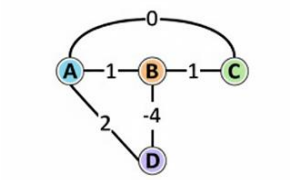

Fig. 1 Sample routing scenario demonstrating the limitations of Dijkstra's algorithm in identifying the energy-efficient paths

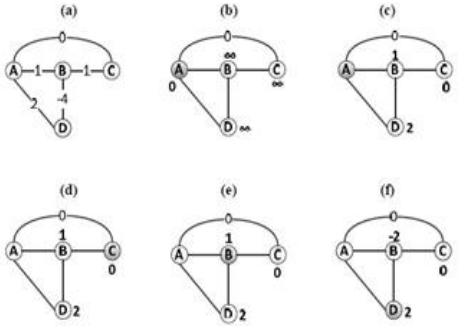

Fig. 2 Illustrative example of Dijkstra's algorithm for a graph incorporating a negative path

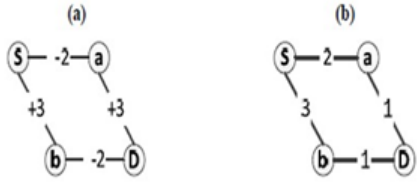

Fig. 3 An example of a basic vehicle routing problem

Recently begun investing heavily in Electric Vehicles (EVs) over the past three to four years. While numerous studies have explored Particle Swarm Optimization (PSO) and its various applications, none have directly addressed the problem of energy-efficient routing for electric vehicles. Some literature has examined energy-efficient routing for electric transports, but then again total of such work remains very limited. Sachenbacher et al. [8] introduced a modified A* search algorithm that accounts for recreating energy capabilities and specific battery (Power Cell) parameters. The resulting design yields a consistent heuristic function with $O(n^2)$ time difficulty. They claimed the algorithm expands each node at most once. It was adapted for battery limitations by modifying the weight function based on potential energy and propulsion energy loss. However, the study did not consider traffic costs, time delays on longer routes, or assume an ideal map environment with uninterrupted travel. Additionally, the $O(n^2)$ growth can be computationally intensive for embedded systems. It gives many route nodes to vehicle drivers. Artmeier et al. [1, 9] explored energy-efficient EV pathing or routing by using graph theory. They designed a general shortest route algorithm with worst-case $O(n^3)$ complexity. Strict constraints ensured the battery does not discharge below minimum levels, while flexible constraints prevented exceeding maximum capacity. Rawashdeh et al. [10] have tried to give examples of how electric vehicles may not be able to use conventional routing methods. The energy-efficient routing problem has been studied metaheuristically. Sweda and Klabjan [11] investigated determining minimum cost paths for EVs lacking sufficient battery charge to reach the destination. They examined backward recursion and approximate dynamic programming, but did not account for regenerative braking. Salehinejad et al. [12] used Ant Colony Optimization (ACO) for route selection, incorporating online and offline traffic data, but applied it to conventional vehicles, not EVs. Siddiqi et al. [13] used PSO for EV route optimization considering recharge time and cost, but did not include regenerative braking. The methodology used in this study applies ACO and PSO metaheuristic optimization techniques. This method determined very reliable, energy-efficient, based EV routes between two (dual) points on a map. This method also considers regenerative braking. The efficiency of ACO and PSO is analysed across various parameters like number of nodes, iterations, time, best cost, and average solution cost on a test database. Y. Shi et. al. [14] examine the engineering and computer science perspectives of Particle Swarm Optimization (PSO), highlighting its developments, applications, and resources. Since its introduction in 1995, the algorithm has evolved through concepts such as constriction factors, inertia weights, and adaptations for dynamic systems. M. Eshani et al. [15] provide an introduction to automotive technology with a focus on battery-operated electric, hybrid (i.e. mixture of both) electric, and fuel cell vehicles. The work is intended for electrical engineers seeking knowledge of automotive systems, as well as automotive engineers aiming to understand electrical propulsion. The purpose of the paper also serves as a useful resource for specialists in related fields, such as electric machinery, who wish to gain deeper insights into vehicle

applications. Rosas-Carrillo et al. [16] have proposed an improved Particle Swarm Optimization (PSO) algorithm called KPSO by introducing an adaptive inertia weight strategy. C. Wang et al. [17] put forward an interaction-driven cooperative evolutionary model with an upgraded ACO for path planning. Fan et al. [18] designed a dual-phase hybrid ACO metaheuristic to resolve routing scenarios involving multiple depots and time varying traffic conditions. The algorithm optimizes routing and charging decisions concurrently while considering traffic-dependent travel times, achieving better solution quality and convergence than traditional ACO. Javadi et al [19] developed a scalable hybrid metaheuristic for EVRP considering better capacity, nonlinear charging and time windows. The framework shows improved scalability and robustness as compared to conventional metaheuristics. Bruni et al. [20] introduced layered graph formulation for EVRP with nonlinear charging functions. The work clearly improves modelling correctness and computational efficiency for true charging behaviour. Song et al. [21] addressed the Heterogenous Energy Replenishment Infrastructures (HERI) governed by several Capacity Constraints (EVRP-HERI-CC). They have engineered a Mixed-Integer-Programming (MIP) Model and A Hybrid-Ant-Colony-Optimization (HACO) framework.

3.1. State-of-the-Art Literature Context (SOT)

New researches focus on advanced SOT methods to solve the Electric Vehicle (EV) pathing or routing problem. These include Deep Reinforcement Learning (DRL) methods to find optimal paths. Among the DRLs Proximal-Policy-Optimization (i.e. PPO) and Deep-Q-Networks (i.e. DQN) are very effective for real time route planning in changing environment. Researchers also widely used hybrid exact methods for complex logistics. Particularly, Branch-and-Price-and-Cut is a leading approach to handle battery constraints and customer time windows. Despite these mathematical advances, classical swarm intelligence algorithms remain highly relevant. Testing methods like ACO and PSO is vital for understanding their performance and establishing reliable baseline solutions for VRP.

4. Metaheuristics

Most real-world optimization challenges tend to exhibit exponential growth in complexity. These problems are often highly nonlinear and can involve types of nonlinearities that are not known beforehand. Due to this inherent complexity, traditional mathematical methods are often inadequate for efficiently solving these problems within reasonable dimensions and time constraints. This has prompted to the creation of innovative techniques capable of addressing complex, high-dimensional optimization issues, resulting in the rise of metaheuristics. Metaheuristic optimization techniques, primarily inspired by natural processes, fall under the category of stochastic algorithms. In 1986, Fred Glover coined the term "metaheuristics" in his Tabu Search. Today, a substantial number of engineering optimization methods rely on metaheuristic algorithms. The variety of developed metaheuristic methods has significantly increased, with well-known examples including PSO, Genetic Algorithms (i.e. GA), Simulated Annealing (i.e. SA), Tabu-Search (i.e. TS), Harmony-Search (HS), Bee-Algorithms (BA), ACO, and Firefly-Algorithms (FA), among others. Solutions generated by metaheuristic approaches do not guarantee optimality, as the complexity of many problems brands it impractical to explore every conceivable solution. Consequently, metaheuristic approaches must balance precision, quality, and accuracy with computational efficiency, both in terms of time and space. The subsequent sections will delve into two specific metaheuristic approaches: ACO and PSO.

4.1. Ant Colony Optimization (ACO) Metaheuristic

This ACO is a bio-inspired (i.e. nature-based) metaheuristic optimization method, which is based on how ants search for food. It is a branch of swarm intelligence. It has been effectively applied to solve complex NP-hard optimization issues. Marco Dorigo initially mentioned ACO in his 1992 PhD thesis [22]. The inspiration for ACO comes from observing how ants determine the quickest path (i.e. route) from a food source to their nest [23]. Ants deposit pheromones on the ground as they move, and other ants select paths with higher pheromone concentrations. This good positive feedback loop causes an ant colony to converge on shortest path. Deneubourg et al [24] conducted experiments in the 1990s to study ant behavior, which served as the foundation for early ACO algorithms. In the experiment called "double bridge", a group of Argentine ants was placed at the edge of two identical bridges. Initially, ants randomly chose either bridge, but over time, the shorter bridge received more pheromones and was selected by most ants. Figure 4 illustrates the geometry of two identical bridges in terms of both structure and length. The authors employed Monte Carlo simulations to investigate how ants select between the two sides of the bridge. Following the crossing of N ants, M units of pheromone are present on side ‘A’ and ‘N – M’ units of pheromone on the bridge side ‘B’. When the next ant reaches the bridge, it has to decide between options ‘A’ and B based on the pheromone levels present. The following ant is more likely to choose branch A if M > N-M. Following this choice, the ant contributes additional pheromones to the selected path.

Over time, due to random fluctuations, one side of the bridge will eventually dominate, leading most ants to favor that path. In contrast, Figure 5 shows a different two-branch bridge where the branches (i.e. paths) are not the same; one path is shorter as compared to the other path. It is clear that branch ‘A’, which is shorter, is the best option for saving time and energy. Initially, when given the option to choose a branch, approximately 50% of the ants select branch A purely just by luck. When ants choose branch A, they reach their food source and return to their nest more quickly, thereby increasing the pheromone levels on that path. As a result, pheromone accumulation on branch A occurs at a faster rate

than on branch B, ultimately leading most ants to prefer branch A. The likelihood of choosing branch 'A' can be expressed as follows:

$$P(A) = \frac{[\tau(A)]^{\alpha} \cdot [\eta(A)]^{\beta}}{[\tau(A)]^{\alpha} \cdot [\eta(A)]^{\beta} + [\tau(B)]^{\alpha} \cdot [\eta(B)]^{\beta}} \quad (1)$$

In this equation:

- P(A) denotes the probability of selecting branch A.
- τ(A) and τ(B) denote the pheromone levels on branches A and B, respectively.
- η(A) and η(B) indicate heuristic information for branches A and B.
- α and β: These are such metrics that find the relative importance of pheromone and heuristic information in the decision-making process.

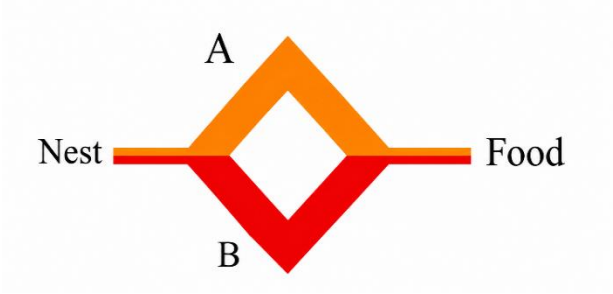

Fig. 4 Bridge branches that are the same

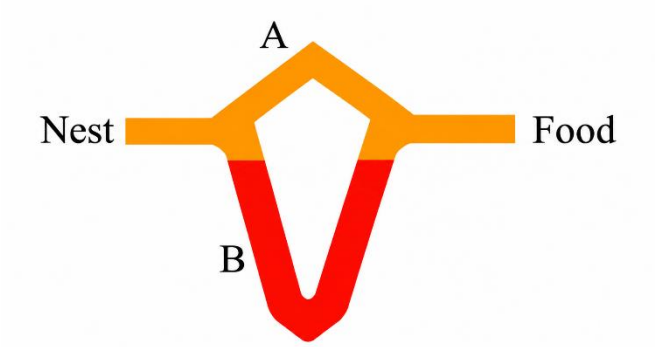

Fig. 5 Various lengths of bridge branches

This formula reflects how the likelihood of selecting a path is influenced by both the pheromone concentration and additional heuristic information, allowing for a balanced decision-making approach in the context of ACO.

Algorithm 1: Ant Colony Optimization (ACO)

1: Objective-function $f(x)$, where $x = (x_1, x_2, \dots, x_n)^T$
2: Initialize pheromone trails τ_{ij} on all edges
3: Set algorithm parameters: α, β, ρ (pheromone evaporation rate), Q (pheromone constant)
4: while termination criterion not met do
5: for each ant k = 1 to m do
6: Place ant k on a starting node (depot)
7: Build solution (route) probabilistically using:

$$P_{ij} = \frac{[\tau_{ij}]^{\alpha} \cdot [\eta_{ij}]^{\beta}}{\sum ([\tau_{il}]^{\alpha} \cdot [\eta_{il}]^{\beta})}$$

8: Ensure capacity and energy constraints satisfied
9: Store constructed route and cost
10: end for
11: Update pheromone trails:
Evaporation: $\tau_{ij} \leftarrow (1-\rho)\tau_{ij}$
Deposit: $\tau_{ij} \leftarrow \tau_{ij} + \Delta\tau_{ij}$,
i. $\Delta\tau_{ij} = \frac{Q}{L_k}$ if edge (i, j) used in route of ant k
12: Find global-optimum-best solution g*
13: end while
14: return the result

Explanation of Variables used in ACO Algorithm

f(x) Objective-function to be minimized (e.g., travel cost, energy consumption, or distance). where $x = (x_1, x_2, \dots, x_n)^T$ is solution vector representing a route through n nodes.

τ_{ij} Pheromone level on edge (i, j) higher values indicate stronger preference.

η_{ij} Heuristic information for edge (i, j), typically chosen as $\eta_{ij} = 1/d_{ij}$ (inverse of distance or cost).

α Pheromone influence parameter controlling the impact of τ_{ij}.

β Heuristic influence parameter controlling the impact of η_{ij}.

ρ Rate of evaporation for Pheromone, $0 < \rho < 1$; prevents limitless accumulation.

Q Pheromone deposit constant, determining how much pheromone is laid on a chosen path.

P_{ij} Probability of selecting edge (i, j) during route construction:

$$P_{ij} = \frac{[\tau_{ij}]^{\alpha} \cdot [\eta_{ij}]^{\beta}}{\sum ([\tau_{il}]^{\alpha} \cdot [\eta_{il}]^{\beta})}$$

L_k Cost of the route constructed by ant k (distance, time, or energy consumption).

$\Delta\tau_{ij}$ Additional pheromone deposited on an edge (i, j) by ant k

$$\Delta\tau_{ij} = \begin{cases} Q/L_k, & \text{if an edge(i,j)is used by ant k} \\ 0, & \text{otherwise} \end{cases}$$

m Total number of ants in the colony (population size)

g* Global best solution (route) found so far across all ants and iterations.

4.2. Particle Swarm Optimization (PSO) Metaheuristic

We have a population based stochastic method within swarm intelligence. This method is known PSO, primarily utilized for resolving continuous non-linear optimization difficulties. It has been developed in 1995 by James Kennedy and Russell Eberhart. They presented this evolutionary

computation method [25]. As suggested by its name, this model is inspired by the natural social behaviour of swarm particles and their interactions. Various organisms, like birds flocking and fish schooling, where individuals pool their experiences to solve complex survival problems. Through a combination of deterministic and stochastic behaviours, individuals learn from one another, ultimately enhancing both personal and group outcomes.

The foundation of Particle Swarm Optimization is built on a straightforward concept. It employs basic mathematical operations, is computationally efficient, requires minimal parameter adjustments, and needs only a few lines of code to implement. PSO is adaptable and can be applied to a diverse range of optimization challenges.

The original work by Kennedy and Eberhart presented a good example demonstrating how Particle Swarm Optimization (PSO) can be utilized to give training to the weights of an artificial neural network. Their proposed algorithm was based on two main components: a nearest neighbour velocity and a random factor termed 'craziness.' Individual agent in swarm was initiated using a random vector for location and velocity. During each iteration, agents attempted to align their velocity with that of their nearest neighbour.

To prevent agents from moving in a uniform direction, a "craziness" factor is introduced, resulting in a more realistic flocking behaviour. Similar to Genetic Algorithms (GA), PSO starts with randomly generated solutions, but unlike genetic algorithms, PSO does not discard solutions based on the idea of the fittest surviving. Instead, it explores the solution space by modifying each agent's trajectory, which creates piecewise paths in a quasi-stochastic manner. Both deterministic and stochastic factors affect how these particles move.

The movement of an individual particle is accelerated towards its own historical best position found in previous iterations as well as the current global best position g*, while also being affected by a stochastic component that allows for random movement. The process can be summarized as follows:

1. Each particle maintains a record of its best-known position and the global optimum best position.
2. Individual particle updates its velocity vector based on its own experience and that of its neighbours.
3. A new position is always calculated using the updated velocity.

This approach results in a dynamic search for optimal solutions, making PSO a powerful tool for various op itemization problems. Figure 6 shows a graphic representation of how a particle-object moves in both directions, its personal best and global best positions.

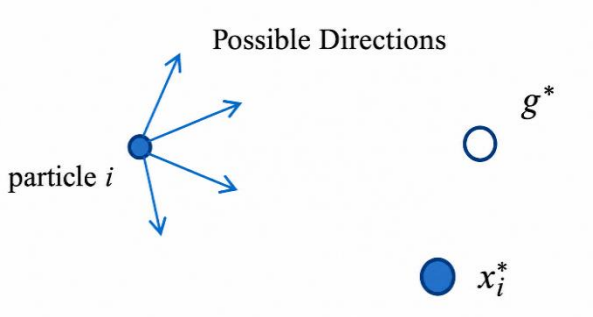

Fig. 6 Pictorial representation of a particle motion in PSO

Let us assume that for a given particle i, $\vec{x}_i$ stands for the location vector and $\vec{v}_i$ (where $\vec{v}_i \in [0, v_{\max}]$) for the velocity vector, the new velocity vector for each iteration of Particle Swarm Optimization (PSO) can be calculated using the following equation:

$$\vec{v}_i^{\,\text{new}} = w \cdot \vec{v}_i^{\,\text{old}} + c_1 \cdot r_1 \cdot (\vec{p}_i - \vec{x}_i) + c_2 \cdot r_2 \cdot (\vec{g} - \vec{x}_i) \quad (2)$$

Where:

- w denotes weight of inertia, which controls the impact of the previous velocity on the recent velocity.
- c_1 is cognitive coefficient and c_2 is social coefficient, which try to find the impact of personal best-position and global best positions.
- Random numbers r_1 and r_2 are evenly distributed across the interval [0, 1].
- p_i is the most well-known location of particle i (personal best).
- g is the best position that any particle in the swarm can find globally.
- x_i denotes current position of particle i.

The equation given above (2) modifies each particle's velocity in each PSO iteration by taking into account its prior velocity, distance from its personal-best position, and distance from the global-best position. The random factors introduce variability in movement, allowing particles to explore the solution space effectively. Once the velocity vector has been modified, the new position of the particle can be calculated as:

$$\vec{x}_i^{\,\text{new}} = \vec{x}_i^{\,\text{old}} + \vec{v}_i^{\,\text{new}} \quad (3)$$

This iterative process continues until a predefined stopping condition is got, like reaching a maximum number of iterations cap or incoming at a satisfying solution.

Algorithm 2: Particle Swarm Optimization (PSO)

1: Objective-function $f(x)$, where $x = (x_1, x_2, \ldots, x_n)^T$
2: Initialize particles with random positions x_i and velocities v_i
3: Set parameters: w (inertia), c_1 (cognitive), c_2 (social)

4: Evaluate $f(x_i)$ for each particle and store personal best p_i
5: Identify global best g^* from all p_i
6: while the termination-criterion is not met Do
7: for each particle i Do
8: update velocity:

$$v_i^{(t+1)} = w \cdot v_i^{(t)} + c_1 \cdot r_1 \cdot (p_i - x_i) + c_2 \cdot r_2 \cdot (g - x_i)$$

9: Update position:

$$x_i^{(t+1)} = x_i^{(t)} + v_i^{(t+1)}$$

10: Evaluate $f(x_i^{(t+1)})$
11. Update personal best p_i
12: end for
13: update global optimum best g^*
14: end while
15: return the results

Explanation of Variables used in PSO Algorithm

f(x) Objective-function to be minimized (e.g., route cost, distance, or energy consumption).
x_i Current position of particle i in the search space (a candidate solution).
v_i Velocity of particle i, which determines how its position is updated in the next iteration.
p_i Personal best-position found so far by particle i.
g∗ Global best-position found so far among all particles in the swarm.
w the inertia weight coefficient, which governs the impact of the preceding velocity on the recent direction vector update.
c_1 A cognitive coefficient, representing the learning factor from a particle's own best position.
c_2 social coefficient, representing the learning factor from the global best solution.
r_1, r_2 Random numbers are uniformly distributed in the range $[0,1]$, which adds stochastic variation.

The velocity of the particle i is updated by using:

$$v_i^{(t+1)} = w \cdot v_i^{(t)} + c_1 \cdot r_1 \cdot (p_i - x_i) + c_2 \cdot r_2 \cdot (g - x_i)$$

The new position is computed as:

$$x_i^{(t+1)} = x_i^{(t)} + v_i^{(t+1)}$$

5. Problem Definition

The core purpose of this work is to find the most energy-efficient path for an Electric Vehicle (EV) traveling from a "Depo" to a designated customer location on a predefined network. The path selection process would be based on four key parameters that influence the decision-making process. These parameters include following factors:

1. Total number of Nodes: The aggregate number of nodes (i.e. number of customers) taken into consideration.
2. Number of Iterations: What number of iterations has been taken into the experiment?
3. Time Taken: Time taken for the specific number of iterations.
4. Best Cost value: This is the solution cost for each iteration and number of nodes in terms of time taken in seconds.

By analysing these parameters, the proposed algorithms (i.e., ACO and PSO) aim to optimize the route selection for EVs, ensuring minimal energy expenditure while effectively reaching the destination. Mathematical formulation for the EVRP with the intention of lessening total travel cost while adhering to capacity constraints: Let us define the following decision variables and parameters used in the mathematical formulation:

5.1. Notations and Decision Variables

- G = (V, E): A kind of directed graph
- V: A complete set of vertices, where 0 represents the depot and $V_c = V \setminus \{0\}$ represents the customer nodes.
- E: It denotes the set of edges.
- $x_{ij} \in \{0,1\}$: A kind of binary decision variable that equals 1 if a EV travels directly from node i to node j, and it becomes 0 otherwise.
- C_{ij} : Total operational (i.e. travel cost) cost or distance when vehicle goes from node i to j.
- d_i: Denotes a payload demand at customer node i
- d_j: Denotes a specific cargo demand requested by customer node j
- d_0: Represents demand at the depot.
- m: Denotes maximum number of operational electric vehicle available within the fleet
- Q: Represents the maximum volumetric payload capacity of the electric vehicle.
- E_{ij}: Energy consumed (positive Value) or regenerated (negative Value), if EV goes from point i to point j.
- B_{max}: The maximum useable battery capacity of the Electric Vehicle (EV).
- u_i: A continuous auxiliary variable denoting the cumulative payload demand up to node i that is used to eliminate subtour
- b_i: A continuous variable that tracks the vehicle State of Charge (SOC) upon arrival at node i.

We can represent this formulation as a Mixed Integer Linear Program (i.e. MILP) as follows:

5.2. Mathematical Model and Constraints

Objective: To minimize total travel cost

$$Minimize \ Z = \sum_{i \in V} \sum_{j \in V, j \neq i} C_{ij} x_{ij}$$

Subject to:

$$\sum_{j \in V, j \neq i} x_{ij} = 1 \quad \forall i \in V_c \tag{5}$$

$$\sum_{i \in V, i \neq j} x_{ij} = \sum_{i \in V, i \neq j} x_{ji} \quad \forall j \in V \tag{6}$$

$$\sum_{j \in V_c} x_{0j} \leq m \quad (7)$$

$$u_j \geq u_i + d_j - Q\left(1 - x_{ij}\right) \quad \forall i, j \in V_c \quad , i \neq j \quad (8)$$

$$d_i \leq u_i \leq Q \quad \forall i \in V_c \quad (9)$$

$$b_j \leq b_i - E_{ij} + B_{\max}\left(1 - x_{ij}\right) \forall i, j \in V_c \, , i \neq j \quad (10)$$

$$0 \leq b_i \leq B_{\max} \quad \forall i \in V \quad (11)$$

$$x_{ij} \in \{0,1\} \quad \forall i, j \in V \quad (12)$$

5.3. Justification of the Mathematical Model

The objective function in Equation (4) reduces the total travel cost or distance across all travelled routes. Equation (5) ensures that each customer is visited exactly once by a vehicle. Equation (6) ensures that if an electric vehicle enters a customer node, it should also leave that customer node. Equation (7) restricts the maximum number of vehicles departing from the central depot to m. Equations (8) and (9) implement Miller-Tucker-Zemlin (MTZ) formulation to eliminate subtours while keeping the vehicle load within capacity Q.

Equations (10) and (11) track the battery SOC between connected nodes based on energy consumption E_{ij}. Negative values of E_{ij} represent regenerative braking, allowing the model to account for energy recovery. Equation (12) talks about binary decision variable that assumes a value of 1 if a Electric Vehicle (EV) goes directly from vertex i to vertex j, and remains 0 if no such path is traversed.

Figure 7 illustrates a simple network map that the metaheuristic algorithms (ACO and PSO) and all simulations would be tested on. The electric vehicle begins from source node 'Depo' and has to reach destination node 'Customers'. The best solution cost in terms of time taken at each iteration and for different number of nodes is listed in Tables 1 and 2.

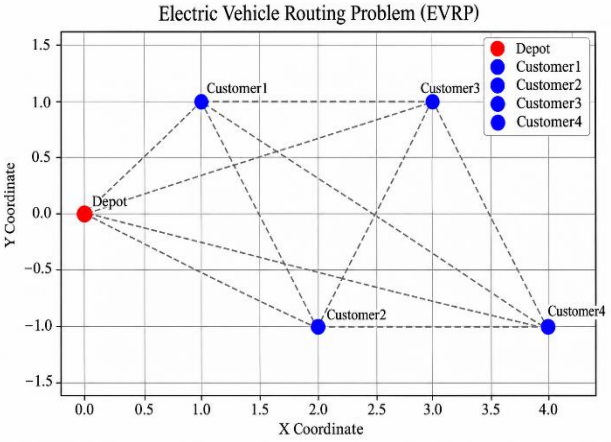

Fig. 7 Map to be applied on the path selection in EVRP for 1 depot and 4 customers

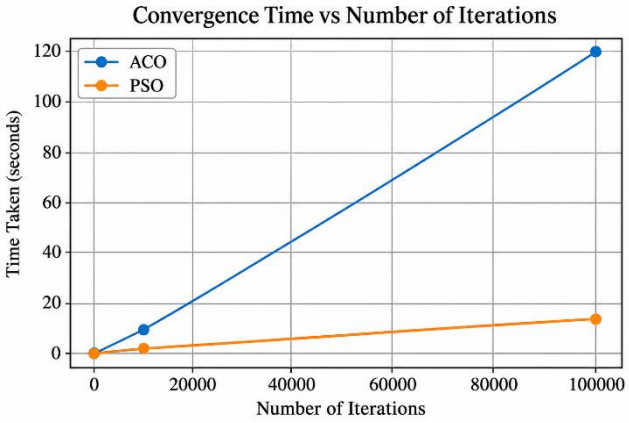

Fig. 8 Convergence time vs number of iteration

6. Experimental Results and Comparison between ACO and PSO

Comparative study of performance evaluation of ACO and PSO with some parameters, as shown within "Figures 8 and 9 and Tables 1 and 2", this is obvious that PSO shows a good better performance in terms of solution optimality and convergence speed as compared to ACO (i.e., average cost and Time taken).

6.1. Experimental Environment and Algorithm Configurations

To ensure that the experiment can be reproduced, all simulations were carried out in Python (Version 3.11). The implementation used the NetworkX library for graph modelling, NumPy for numerical computations, and Matplotlib for visualization. Entire computational simulations were done on a workstation, which is well equipped with an 11th-Generation Intel Core i3 processor, 8 GB of RAM, and a 64-bit Windows 11 OS.

Synthetic benchmark maps with 5 to 25 customers node and one depot were generated on bounded 2D coordinate plane. Negative values in the energy matrix C_{ij} were used to represent regenerative breaking. The hyperparameter settings used for all simulation runs are listed in Table 3.

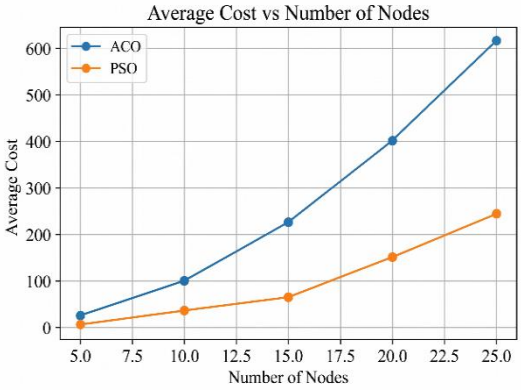

Fig. 9 Average cost vs number of nodes

6.2. Enhanced Result Analysis and Benchmarking

A comparison of the results obtained within "Tables 1 and 2" shows clear performance differences between ACO and PSO across all test cases.

6.2.1. Computational Efficiency and Algorithmic Speedup

The results show that PSO requires less computational time than ACO as the problem size increases.

a) Execution Time: For the 5-customer instance with 100,000 iterations, ACO took 118.06 seconds, while PSO completed the same task in 14.05 seconds, providing an 8.40 times speedup.
b) Scalability: As the problem size increased to 25 customer nodes, ACO took 927.49 seconds, while PSO required only 45.05 seconds, resulting in a 20.59 times computational speedup.
c) Reason for the Speedup: The faster execution of PSO is due to its simple vector-based updates, whereas ACO repeatedly computes transition probabilities for route construction. So, ACO increases computational cost.

6.2.2. Solution Quality and Convergence Behaviour

The obtained results in "Figures 8 and 9, and Tables 1 and 2" clearly indicate that PSO finds more energy-efficient routes than ACO. Cost Reduction: For the 10-customer benchmark, PSO obtained a routing cost of 20 compared with 100 for ACO. So, PSO reduces the routing cost by 80%.

Large Scale Performance: For the 25-customer nodes, PSO achieved a routing cost of 275 compared with 584 for ACO, thus representing a 52.91% reduction in routing cost.

Convergence Behaviour: As shown in Table 1, ACO often converges early with little or no improvement in routing cost after the initial iterations. This suggests that ACO can become trapped in local optima because of pheromone reinforcement.

On the other hand, PSO maintains a better balance between exploration & exploitation through its cognitive coefficients (c_1) and social (c_2) coefficients.

Table 1. Performance evaluation results for different nodes and iterations using ACO

No. Of Nodes	No. of Iterations	Times Taken(s)	Best Cost Value	Average Solution Cost
5	100	0.24339795112609863 seconds	25	25
	1000	0.9008190631866455 seconds	25	
	10000	9.344828128814697 seconds	25	
	100000	118.05595207214355 seconds	25	
10	100	0.5896496772766113 seconds	100	100
	1000	2.0705132484436035 seconds	100	
	10000	25.30476474761963 seconds	100	
	100000	235.7899513244629 seconds	100	
15	100	0.25013136863708496 seconds	225	225
	1000	2.775852680206299 seconds	225	
	10000	31.38131356239319 seconds	225	
	100000	347.06724977493286 seconds	225	
20	100	0.5841991901397705 seconds	400	400
	1000	3.5951240062713623 seconds	400	
	10000	35.75638818740845 seconds	400	
	100000	373.9925739765167 seconds	366	
25	100	0.5091691017150879 seconds	625	614
	1000	5.116662502288818 seconds	625	
	10000	80.57208800315857 seconds	625	
	100000	927.4948444366455 seconds	584	

6.3. Scalability, Limitations, and Practical Implications

a) Scalability Analysis: PSO shown better results as compared to ACO on the tested 5 to 25 customer nodes. PSO scaled more efficiently when problem size increases. Still, due to the fast growth of the search space, both algorithms face challenges as they deal with large urban networks. As ACO has repeated probability updates, so it becomes computationally expensive. On the other hand, PSO might lose efficiency in handling complex discrete constraints. These limitations suggest the need for hybrid methods or graph-partitioning techniques to improve scalability.

b) Research Limitations: This work assumes a parallel EV fleet with similar battery characteristics and operating conditions. The proposed model uses static edge weights. It does not consider random traffic condition such as congestion or signal delays. It also represents regenerative breaking using simplified energy recovery values without considering reduced charging efficiency at high battery State-of-Charge (SOC) levels.

c) Practical Implications: This framework remains highly useful despite its current limitations. Its computational efficiency calculates paths quickly. This speed allows for real time EV navigation and fleet management. Energy-efficient routes also lower overall electricity use. It eases charging demand and improves overall efficiency of electric vehicle operations.

Table 2. Performance evaluation results for different nodes and iterations using PSO

No. Of Nodes	No. of Iterations	Times Taken(s)	Best Cost Value	Average Solution Cost
5	100	0.008638620376586914 seconds	5	7
	1000	0.125138521194458 seconds	7	
	10000	1.2974236011505127 seconds	10	
	100000	14.049926996231079 seconds	6	
10	100	0.010138273239135742 seconds	35	36
	1000	0.19927573204040527 seconds	40	
	10000	2.033184051513672 seconds	50	
	100000	21.3650860786438 seconds	20	
15	100	0.01980447769165039 seconds	45	64
	1000	0.26328158378601074 seconds	64	
	10000	2.7972588539123535 seconds	90	
	100000	29.096723079681396 seconds	60	
20	100	0.03786945343017578 seconds	120	150
	1000	0.3474888801574707 seconds	160	
	10000	3.498270034790039 seconds	200	
	100000	35.42889904975891seconds	120	
25	100	0.04051947593688965seconds	146	242
	1000	1.0888397693634033seconds	200	
	10000	4.2542877197265625seconds	350	
	100000	45.04936742782593seconds	275	

Table 3. Fixed metaheuristic hyperparameters value for reproducibility

Algorithms Used	Hyperparameter	Symbol	Assigned Value
Ant Colony Optimization (ACO)	Population Size (Ants)	m	20
	Pheromone Influence Factor (PIF)	α	1.0
	Heuristic Influence Factor (HIF)	β	2.0
	Evaporation Rate (ER)	ρ	0.1
	Pheromone Constant	Q	100
Particle Swarm Optimization (PSO)	Swarm Size (Particles)	m	20
	Inertia Weight (IW)	w	0.5
	Cognitive Coefficient (CC)	c_1	1.5
	Social Coefficient (SC)	c_2	1.5
	Maximum Velocity Bound	$v_{\max}$	4.0

Table 3 presents the exact hyperparameters used for both ACO and PSO algorithms throughout the experimental study. These parameters values were kept fixed across all instances to ensure fair comparison and reproducibility, and evaluation of algorithmic performance. The following are the optimized routes for different numbers of nodes.

6.4. Optimized Route Visualizations Using ACO and PSO

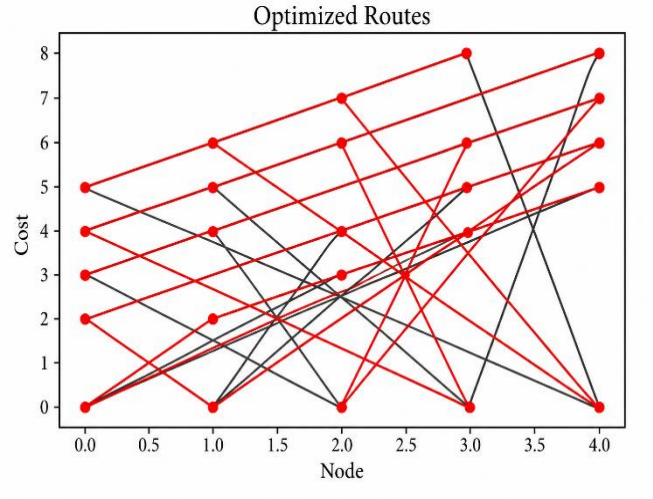

Fig. 10 ACO with its best solution

Best solution: [0,1,4,2,3,0], Best solution cost: 25

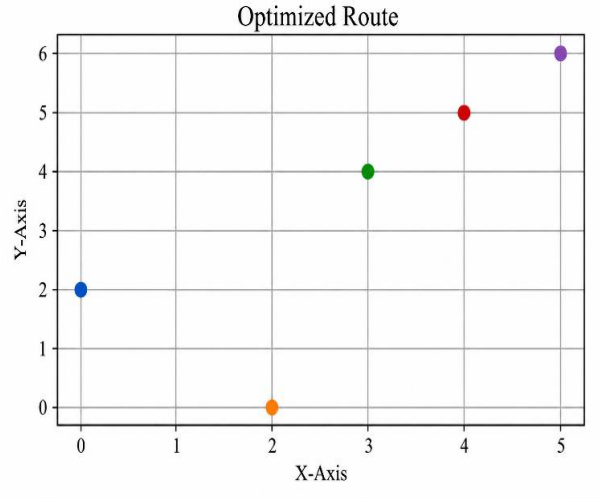

Fig. 11 PSO with its best solution

Global Best solution: [1,1,1,1,4], Best solution cost: 7

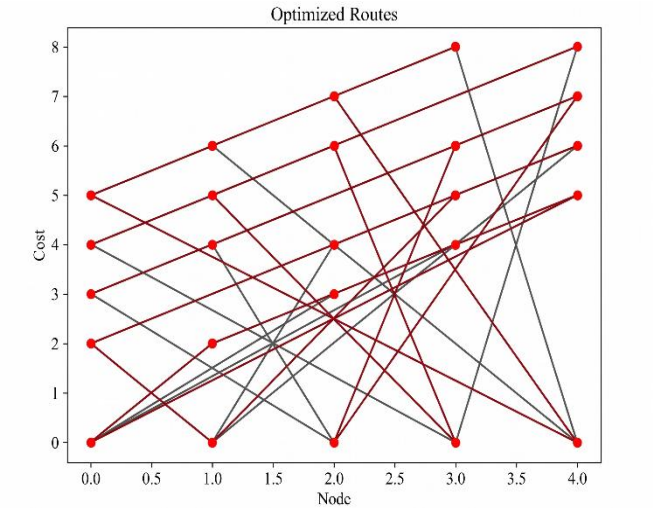

Fig. 12 Number of nodes: 5

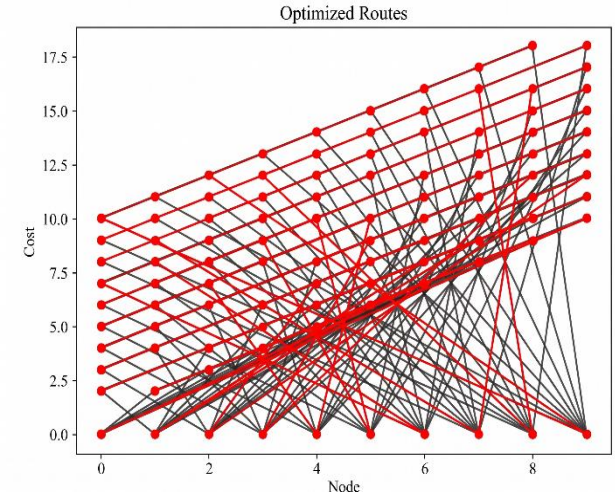

Fig. 13 Number of nodes: 10

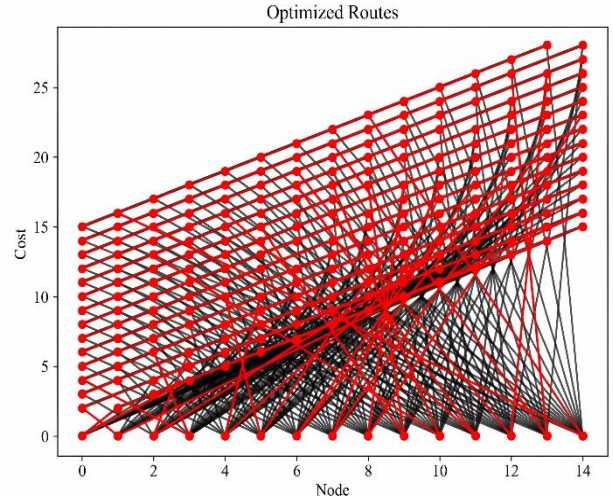

Fig. 14 Number of nodes: 15

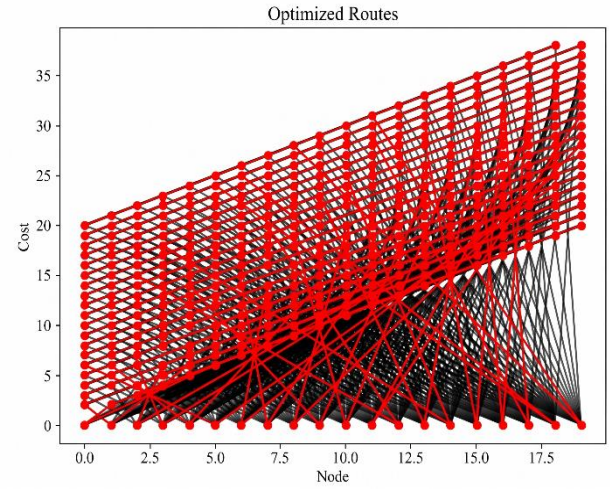

Fig. 15 Number of nodes: 20

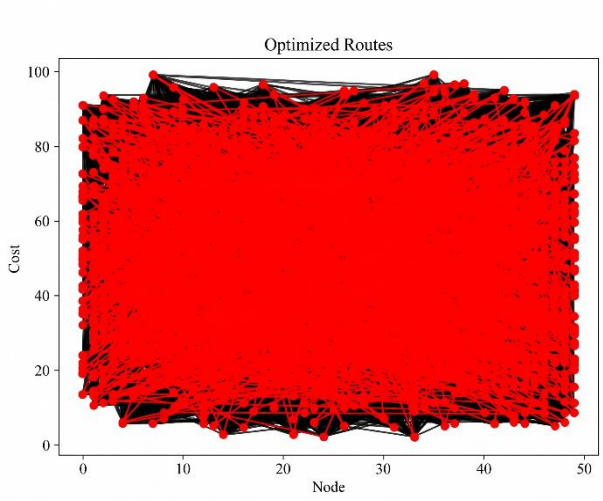

Fig. 16 Optimized routes for 50 nodes using ACO

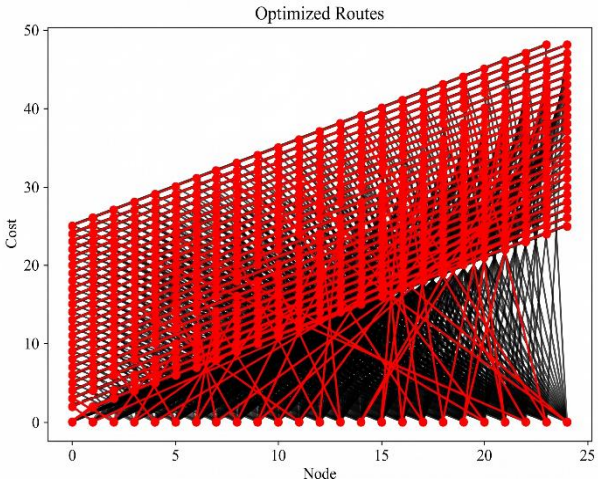

Fig. 17 Optimized routes for 25 nodes using ACO

Table 4. Comparison of variables and their roles in ACO and PSO

Variables	ACO	PSO
$f(x)$	Objective function (e.g., distance, energy, or cost of route).	Objective function (e.g., distance, energy, or cost of route).
x	Route constructed by ants, represented as a sequence of nodes.	Position of a particle in search space, representing a candidate solution.
τ_{ij}	Pheromone intensity on edge (i,j), influences route choice.	Not applicable.
η_{ij}	Heuristic value of edge (i,j), typically $1/d_{ij}$ (inverse distance).	Not applicable.
α	Weighting factor of pheromone trail.	Not applicable.
β	Weighting factor of heuristic information.	Not applicable.
ρ	Rate of Pheromone evaporation, prevents unlimited accumulation.	Not applicable.
Q	Pheromone deposit constant controlling reinforcement strength.	Not applicable.
P_{ij}	Probability of selecting edge (i,j) based on τ_{ij} and η_{ij}.	Not applicable.
L_k	Cost of route constructed by ant k.	Not applicable.
$\Delta\tau_{ij}$	Pheromone deposition: Q/L_k if an Edge (i,j) used in the path of an ant.	Not applicable.
$\Delta\tau_{ij}$	Pheromone deposition: Q/L_k if an Edge (i,j) used in the path of an ant.	Not applicable.
m	Total population size of ants in the colony.	Absolute No. of particles within swarm.
g^*	Global best route found across all ants.	Global best position found across all particles.
p_i	Not applicable.	Personal best position found by particle i.
v_i	Not applicable.	Velocity of particle i, controls movement in search space.
w	Not applicable.	Inertia weight, balances exploration vs. exploitation.
c_1, c_2	Not applicable.	Acceleration coefficients: c_1 (cognitive), c_2 (social).
r_1, r_2	Not applicable.	Random numbers $\in [0,1]$, introduce stochasticity in velocity update.

Table 4 shows the primary variables used in both the ACO and PSO algorithms. It highlights their respective role in optimisation process. This comparison provides a unified understanding of all algorithm-specific notations. Finally, this

clear breakdown makes it much easier to interpret mathematical formulations

7. Conclusion and Future Scope

This study conducted a comparative performance evaluation of PSO and ACO across various metrics, including the number of nodes, iterations, and time taken to achieve optimal solutions.

The findings, as illustrated in "Figure 8 and 9" and detailed in "Table 1 and 2", clearly demonstrate that PSO outperforms ACO in several critical aspects.

Key Findings:

1. Convergence Speed: PSO consistently achieved faster convergence rates compared to ACO. This indicates that PSO is more efficient in navigating the solution space, leading to quicker identification of optimal solutions.

2. Solution Optimality: The average cost associated with solutions derived from PSO was significantly lower than that of ACO. This highlights PSO's effectiveness in not only finding solutions faster but also ensuring those solutions are of higher quality.

3. Time Efficiency: The time taken by the PSO algorithm to reach the best solution was considerably less than that of ACO. This found efficiency is mainly beneficial in applications where computational resources and time are critical factors. Implications: The superior performance of PSO over ACO suggests that, for applications requiring rapid and optimal solutions, particularly in complex environments with multiple nodes, PSO should be favoured. This research contributes valuable insights into strengths and weaknesses of these optimization approaches, guiding practitioners in selecting the most appropriate method for their specific needs.

Future Research

Further studies could explore hybrid approaches that combine the strengths of both ACO and PSO to enhance performance even further. Additionally, investigating the impact of varying parameters on algorithm performance could provide deeper insights into their operational dynamics. In conclusion, our findings affirm that PSO is a more effective optimization technique than ACO under the conditions tested, paving the way for its broader application in various fields requiring efficient problem-solving capabilities.

Funding Statement

Author declares that no financial support, institutional grants, or other support were received for this manuscript. Compliance with Ethical Standards

Conflicts of Interest

Authors declare that there is no conflict of interest that influence the objective of the research.

Ethical Compliance

This article presents purely computational or theoretical research. This article does not involve any studies with human participants performed by any of the authors.

References

[1] Andreas Artmeier et al., "The Optimal Routing Problem in the Context of Battery-Powered Electric Vehicles," *CPAIOR Workshop on Constraint Reasoning and Optimization for Computational Sustainability*, pp. 1-13, 2010. [Google Scholar] [Publisher Link]

[2] Peter J. Fontaine, "Shortening the Path to Energy Independence: A Policy Agenda to Commercialize Battery-Electric Vehicles," *The Electricity Journal*, vol. 21, no. 6, pp. 22-42, 2008. [CrossRef] [Google Scholar] [Publisher Link]

[3] Nora Touati-Moungla, and Vincent Jost, "Combinatorial Optimization for Electric Vehicles Management," *Journal of Energy and Power Engineering*, vol. 6, no. 5, pp. 738-743, 2012. [CrossRef] [Google Scholar]

[4] Richard Bellman, "On a Routing Problem (No. Rand-P 1000)," *Rand Corp Santa Monica Ca*, 1956. [Google Scholar]

[5] Rami Abousleiman, Abdullah Al-Refai, and Osamah Rawashdeh, "Charge Capacity Versus Charge Time in CC-CV and Pulse Charging of Li-Ion Batteries," *SAE 2013 World Congress and Exhibition*, Detroit, Michigan, United States, 2013. [CrossRef] [Google Scholar] [Publisher Link]

[6] E.W. Dijkstra, "A Note on Two Problems in Connexion with Graphs," *Edsger Wybe Dijkstra: His Life, Work, and Legacy*, pp. 287-290, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[7] L.R. Ford Jr., and D.R. Fulkerson, "A Simple Algorithm for Finding Maximal Network Flows and an Application to the Hitchcock Problem," *Canadian Journal of Mathematics*, vol. 9, pp. 210-218, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[8] Martin Sachenbacher et al., "Efficient Energy-Optimal Routing for Electric Vehicles," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 25, no. 1, pp. 1402-1407, 2011. [CrossRef] [Google Scholar] [Publisher Link]

[9] Andreas Artmeier et al., "The Shortest Path Problem Revisited: Optimal Routing for Electric Vehicles," *Annual Conference on Artificial Intelligence*, Karlsruhe, Germany, pp. 309-316, 2010. [CrossRef] [Google Scholar] [Publisher Link]

[10] Rami Abousleiman, and Osamah Rawashdeh, "Energy-Efficient Routing for Electric Vehicles using Metaheuristic Optimization Frameworks," *MELECON 2014 - 2014 17th IEEE Mediterranean Electrotechnical Conference*, Beirut, Lebanon, pp. 298-304, 2014. [CrossRef] [Google Scholar] [Publisher Link]

[11] Timothy M. Sweda, and Diego Klabjan, "Finding Minimum-Cost Paths for Electric Vehicles," *2012 IEEE International Electric Vehicle Conference*, Greenville, SC, USA, pp. 1-4, 2012. [CrossRef] [Google Scholar] [Publisher Link]

[12] Hojjat Salehinejad, Farhad Pouladi, and Siamak Talebi, "A New Route Selection System: Multiparameter Ant Algorithm based Vehicle Navigation Approach," *2008 International Conference on Computational Intelligence for Modelling Control and Automation*, Vienna, Austria, pp. 1089-1094, 2008. [CrossRef] [Google Scholar] [Publisher Link]

[13] Umair Farooq Siddiqi, Yoichi Shiraishi, and Sadiq M. Sait, "Multi-Constrained Route Optimization for Electric Vehicles (EVs) using Particle Swarm Optimization (PSO)," *2011 11th International Conference on Intelligent Systems Design and Applications*, Cordoba, Spain, pp. 391-396, 2011. [CrossRef] [Google Scholar] [Publisher Link]

[14] Eberhart, and Yuhui Shi, "Particle Swarm Optimization: Developments, Applications and Resources," *Proceedings of the 2001 Congress on Evolutionary Computation*, Seoul, Korea, vol. 1, pp. 81-86, 2001. [CrossRef] [Google Scholar] [Publisher Link]

[15] Mehrdad Ehsani et al., *Modern Electric, Hybrid Electric, and Fuel Cell Vehicles*, 3rd ed., CRC press, 2018. [Google Scholar] [Publisher Link]

[16] Ary Shared Rosas-Carrillo et al., "UAV Path Planning using an Adaptive Strategy for the Particle Swarm Optimization Algorithm," *Drones*, vol. 9, no. 3, pp. 1-17, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[17] Shouliang Zhu, and Chao Wang, "An Interaction-Enhanced Co-Evolutionary Algorithm for Electric Vehicle Routing Optimization," *Applied Soft Computing*, vol. 165, pp. 1-16, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[18] Lijun Fan, "A Two-Stage Hybrid Ant Colony Algorithm for Multi-Depot Half-Open Time-Dependent Electric Vehicle Routing Problem," *Complex and Intelligent Systems*, vol. 10, no. 2, pp. 2107-2128, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[19] Hatef Javadi et al., "Flexible and Scalable Solutions for Electric Vehicle Routing Problems: A Hybrid Metaheuristic Framework," *IEEE Access*, vol. 13, pp. 217519-217541, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[20] Maria Elena Bruni, Maximiliano Cubillos, and Ola Jabali, "Layered Graph Models for the Electric Vehicle Routing Problem with Nonlinear Charging Functions," *Networks*, vol. 85, no. 1, pp. 113-126, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[21] Bowen Song, and Rui Xu, "Electric Vehicle Routing Problem with Heterogeneous Energy Replenishment Infrastructures under Capacity Constraints," *Algorithms*, vol. 18, no. 4, pp. 1-35, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[22] Marco Dorigo, "*Optimization, Learning and Natural Algorithms*," Ph.D. Thesis, Politecnico di Milano, 1992. [Google Scholar]

[23] Marco Dorigo, Mauro Birattari, and Thomas Stutzle, "Ant Colony Optimization," *IEEE Computational Intelligence Magazine*, vol. 1, no. 4, pp. 28-39, 2006. [CrossRef] [Google Scholar] [Publisher Link]

[24] J. -L. Deneubourg et al., "The Self-Organizing Exploratory Pattern of the Argentine Ant," *Journal of Insect Behavior*, vol. 3, no. 2, pp. 159-168, 1990. [CrossRef] [Google Scholar] [Publisher Link]

[25] James Kennedy, and Russell C. Eberhart, "Particle Swarm Optimization," *Proceedings of ICNN'95 - International Conference on Neural Networks*, Perth, WA, Australia, vol. 4, pp. 1942-1948, 1995. [CrossRef] [Google Scholar] [Publisher Link]