I. J. Engineering and Manufacturing, **2025, 5, 31-45**
Published Online on October 8, 2025 by MECS Press (http://www.mecs-press.org/)
DOI: 10.5815/ijem.2025.05.03

ACO-QL: Enhancing ACO Algorithm for Routing in MANETs Using Reinforcement Learning

Yahia Mohsen Abu Saqer*
Islamic University of Gaza/Faculty of IT, Gaza, P850, Palestine
E-mail: yahiasaqer@gmail.com
ORCID ID: https://orcid.org//0009-0007-4965-0468
*Corresponding author

Khalil Mohammed Eslayyeh
Islamic University of Gaza/Faculty of IT, Gaza, P850, Palestine
E-mail: Khalil.eslayyeh@gmail.com
ORCID ID: https://orcid.org//0009-0006-2843-438X

Nasser Majed Abudalu
Islamic University of Gaza/Faculty of IT, Gaza, P850, Palestine
E-mail: Abudalunasser@gmail.com
ORCID ID: https://orcid.org//0009-0004-8991-7399

Aiman A. Abusamra
Islamic University of Gaza/Faculty of Engineering, Gaza, P850, Palestine
E-mail: aasamra@iugaza.edu.ps
ORCID ID: https://orcid.org//0000-0003-4652-3993

Received: 17 February, 2025; Revised: 14 March, 2025; Accepted: 17 April, 2025; Published: 08 October, 2025

Abstract: ACO-based routing protocols like AntHocNet have emerged as a solution for adaptive routing in MANETS. Likewise, deep Q-learning based protocols are suitable for complex and dynamic environments like MANETs and utilizing real time data for better decision-making. However, there is lack of studies in enhancing ACO-based protocols using Q-learning in a new hybrid protocol, and comparing it with the established ACO-based protocol AntHocNet.

By combining ACO's strengths (eg. Multi-agent pathfinding and historical data creatd by pheromones) and combine it with key components of Q-learning, then we have a promising protocol ready to be compared with AntHocNet. Previous studies have explored integrating ML with MANET routing, but few of them, if any, have explored enhancing ACO using ML techniques. Therefore, we propose two new protocols: ACO-QL and ACO-DQN.

One uses Q-learning and the latter uses deep Q-learning. After conducting many experiments by running implementations of ACO-DQN, ACO-QL, and AntHocNet on a MANET simulation, we found out that AntHocNet is superior to ACO-DQN in terms of execution time, end-to-end delay, and path cost in most cases, but on the other hand ACO-DQN achieved better packet delivery ratio and throughput results. Meanwhile, ACO-QL consistently achieved lower packet delivery ratios than AntHocNet, and mostly matching AntHocNet's performance in terms and of other metrics, making it a valid lightweight and faster alternative.

Index Terms: MANET, Machine Learning, Reinforcement Learning, Deep Learning, Q-Learning, ACO, Deep Q-Learning

1. Introduction

The rapid evolution of Mobile Ad Hoc Networks (MANETs) has driven the demand for efficient and adaptive routing protocols that can dynamically respond to varying network conditions, as they are formed spontaneously, making them suitable in urgent situations, decentralized, and provide dynamic topology, as nodes constantly move and change [1]. With the inherent challenges of high mobility, frequent topology changes, and limited resources, traditional routing protocols often fall short of ensuring optimal performance [2].

Ant Colony Optimization (ACO) is an optimization algorithm inspired by the behavior of ants searching for food and is a part of the metaheuristic algorithms used to find solutions to complex problems [3]. As ants take the route with the most pheromone deposited by other ants that is most likely to be the optimal route, this iterative methodology can be applied to MANETs, by placing artificial ants in the network, each ant constructs a solution step by step by moving from one node to another based on the pheromone levels on each route and problem-specific heuristic information, once the artificial ants complete their paths, the pheromone levels are updated on each route, as the pheromone levels are either increased or decreased (evaporate). The ACO algorithm has emerged as a promising solution for adaptive routing in MANETs in the form of routing protocols like the AntHocNet protocol, it is established that AntHocNet has outperformed ADOV on different evaluation criteria such as average end to end delay, throughput, and packet delivery ratio [4], and that its reactive-proactive strategy is suitable for to adapt quickly to network changes as in MANETs. Similarly, AntHocNet is established as the best performing ACO-based protocol in terms of packet delivery ratio, and throughput, because it has a higher probability of exploring new paths, it offers greater adaptability, but this comes at a higher cost and requires more resources to implement due to the significant amount of ant traffic generated [5].

However, existing ACO-based protocols primarily rely on parameters that could benefit from being more adaptive and dynamic, as traditional ACO uses pheromone evaporation and heuristic values to guide ants, but these mechanisms are not dynamically adjusted based on network feedback, while ACO has proven to be one of the most effective algorithms for solving NP-hard problems such as the Traveling Salesman Problem (TSP), as numerous strategies have been developed to enhance ACO, relatively few studies have focused on tuning its parameters, which have a direct impact on the algorithm's performance [6]. Additionally, pure ACO floods the network with ants for route discovery, leading to high overhead. The ant traffic overhead is a known issue in AntHocNet as we pointed out earlier [4]. So, this leaves us with two challenges regarding ACO-based routing protocols, high overhead, and the need for more dynamic tuning.

By integrating Q-learning, the hybrid model could prioritize known high-quality paths (exploitation) while using ACO for occasional exploration, reducing redundant ant propagation. ACO-based protocols sometimes struggle with convergence to an optimal solution in dynamic and large-scale networks. The Q-learning component can help guide the system to faster convergence by learning which paths yield better results over time, thus reducing reliance on random exploration alone.

To address this challenge, researchers have explored integrating ML techniques with MANET routing protocols to enhance the adaptability, or by enhancing the ACO algorithm using techniques like cluster head selection and network information analysis. While prior studies have shown potential, outside of few protocols like AntHocNet, there is still a lack of comprehensive approaches that utilize real-time data predictions to improve the performance of ACO-based routing protocols for better decision-making, there are even few studies that explored the integration of ML techniques with the ACO algorithm. Additionally, most of these studies rarely addressed the issues of high overhead and sometimes, they lack applicability on realistic MANET simulations.

This paper proposes a novel hybrid routing protocol that integrates ACO with the Q-learning algorithm which is a reinforcement learning algorithm, to make the ACO algorithm more adaptive, especially in complex and dynamic environments, the nature of the Q-learning algorithm as a model-free algorithm (does not need any prior knowledge about the environment) makes it a suitable option for enhancing the ACO algorithm, as it encourages the ants to focus on exploring optimal paths as the rewards given by the Q-learning algorithm increases the chances of the ants to take these paths, by dynamically adjusting the weight of pheromones, energy efficiency, and path costs during runtime. This allows the protocol to learn optimal parameter configurations as the network evolves. Making the ACO algorithm has more dynamic parameter tuning. This combination allows the algorithm to balance immediate rewards (e.g., low delay) with long-term path stability, a capability not fully realized in standalone ACO or RL approaches. As for overhead and performance, the Q-learning component can help guide the system to faster convergence by learning which paths yield better results over time, thus reducing reliance on random exploration alone. The protocols are simulated using the NetworkX Python library for the creation, manipulation, and study of the structure, dynamics, and functions of complex networks and the Matplotlib visualization library to demonstrate the results and the comparison between the enhanced ACO with Q-learning (ACO-QL) against the well established AntHocNet protocol. And as MANETs can become more complex, another algorithm that combines ACO with Deep Q-learning (ACO-DQN) is implemented and compared to the two aforementioned algorithms.

The goal is to demonstrate the effectiveness of the proposed protocols in improving routing efficiency in MANET environments under diverse conditions, and enhancing the adaptability and efficiency of ACO-based routing protocols.

The paper's contributions are:

- Developing a simulation of MANETs using the NetworkX and Matplotlib libraries. Which can be used for research purposes, lightweight, and accessible as it is implemented in Python.
- Implementing the ACO-QL, ACO-DQN, and AntHocNet on the aforementioned MANET simulation, and comparing their performances.

2. Related Work

Routing in dynamic and resource-constrained network environments, such as MANETs, VANETs, and network operations center (NoC) systems, has witnessed significant advancements driven by the integration of machine learning (ML) and optimization techniques. The primary focus of recent research has been on improving performance metrics like throughput, latency, energy efficiency, and packet delivery ratio by leveraging reinforcement learning (RL), heuristic-based algorithms, and hybrid frameworks. This section reviews related works grouped into common research directions, providing a foundation for integrating Q-learning with Ant Colony Optimization (ACO) to enhance routing.

Several studies have explored the use of ML techniques to optimize routing in wireless ad-hoc networks. S. Kaushik, K. Tripathi, R. Gupta and P. Mahajan (2021) provided a comprehensive analysis of ML's role in enhancing MANET and VANET protocols, highlighting its ability to improve throughput, packet delivery ratios, and QoS [7]. However this work lacks necessary technical details (e.g. clarifying the implementations) and it lacks empirical validation as it relies on comparing previous works with possible varying assumptions and conditions, it becomes hard to draw a definitive conclusion. Similarly, Mohammad Arif, Bhargavi, Swaroopa, Karuturi, and Balamurugan, A. (2024) demonstrated the superiority of deep reinforcement learning methods like DQN in improving energy efficiency and communication reliability, achieving a 96% packet delivery ratio and minimal latency which is better than ML techniques that were compared with DQN in that research, namely, MARL, and SOM [8]. However, DQN introduced additional overhead, which was acknowledged in the study, but not addressed. These studies emphasize the growing potential of ML to address the dynamic and resource-intensive challenges of ad-hoc networks, but both works ignored swarm intelligence and its advantages or any novel approach that could improve on existing literature.

In VANETs, heuristic-based approaches have also shown promise. Akhilesh Bijalwan, Iqram Hussain, Kamlesh Chandra Purohit, and M. Anand Kumar (2023) introduced an enhanced ACO framework leveraging dynamic transmission ranges and improved cluster head selection, significantly boosting network stability and scalability as this novel approach of enhancing cluster head selection using an iterative process that employs ACO is better than traditional ACO when it comes to packet loss ratio, delay, throughput packet delivery, and clustered nodes lifetimes [9]. However, the throughput decreases as the number of nodes increase, and the study was inducted in a relatively small scale of 100 vehicles in a 4 square kilometers area, which may suggest challenges in scalability, Zahid Khan et al. (2020) proposed a street-centric routing scheme combining multipath routing and ACO-based clustering, reducing computational costs and end-to-end delays [10]. Both studies underscore the adaptability of ACO algorithms when combined with intelligent clustering and routing mechanisms. Although these studies target VANETs, adapting them to MANETs can be challenging due to structural difference between MANETs and VANETs, as MANETs can have more random structure unlike VANETs' structured movement that follows lanes, roades and predictable speeds. Scalability can be a major challenge too, as MANETs can be larger and sparser than VANETs.

Reinforcement learning has emerged as a powerful tool for dynamic routing optimization as shown in [8], and Zoubir Mammari (2019) provided a foundational review of RL-based routing protocols, focusing on their adaptability and QoS improvements [11]. Saeed Kaviani et al. (2021) Developed DeepCQ+, a MADRL-based routing protocol for MANETs that integrates Q-learning with deep neural networks. This hybrid approach demonstrated superior scalability and robustness, maintaining high throughput while minimizing overhead [12]. This work is the closest one to our study, but it lacks addressing scalability, as it is concerned with $\leq$30 nodes, while real-world MANETs often involve more nodes. Performance in large-scale or ultra-dense networks remains unverified. It negelcts addressing additional overheads and simulating heterogeneous MANETs.

In NoC systems, reinforcement learning has also been explored for adaptive routing. Hsin, Chang, Su and Wu (2014) introduced the Network Information Region (NIR) framework, enabling the integration of spatial and temporal information in routing. The proposed ACO-PhD algorithm achieved significant performance improvements by combining pheromone diffusion with network information analysis [13]. This work, alongside with [9], are the only works known to us that tried to enhance the traditional ACO with approaches that are not related to RL.

The fusion of ML and optimization techniques has proven effective in addressing complex routing challenges. R. Amin et al. (2021) surveyed the application of supervised, unsupervised, and RL techniques for SDN routing, identifying gaps in reproducibility and practical implementation [14]. Davi Ribeiro Militan et al. (2021) introduced e-RLRP, a reinforcement learning-based protocol that dynamically adjusts Hello message intervals to reduce overhead and optimize performance [15]. However, the evaluation prioritizes VoIP via the E-model. Performance for other traffic types is not assessed, limiting generalizability claims, it ignores energy efficiency too, and it lacks adaptive parameters tuning. These studies emphasize the need for standardized frameworks and hybrid solutions that integrate ML with traditional routing algorithms.

When it comes to combining RL with ACO, there are few works that have tackled this topic. Satyanarayana S. Nimmala et al. (2024). Here the paper presents Dynamic RL-ACO, a load balancing approach for cloud computing that combines ACO and RL. RL predicts real-time load distribution strategies, while ACO dynamically adjusts task allocation. CloudSim simulations using Google Cluster Data show that Dynamic RL-ACO outperforms traditional algorithms, achieving a 15% reduction in response time, 20% improvement in resource utilization, 25% faster adaptation to workload changes, and 15% reduction in energy consumption. These results highlight the method's scalability, adaptability, and effectiveness in modern cloud environments [16]. Ghanshyam Prasad Dubey et al. (2023) present the PPO-ACO algorithm for optimal path selection in Wireless Sensor Networks (WSN) used in the Internet of Things (IoT). It combines Proximal Policy Optimization (PPO) from reinforcement learning and Ant Colony Optimization (ACO) to address the trade-off between energy efficiency and security. PPO learns the path selection policy, while ACO guides the search with pheromone updates. Simulation results show PPO-ACO outperforms existing algorithms in terms of active nodes and residual energy, indicating higher efficiency. The approach is suitable for IoT applications, particularly in 5G communications [17]. S. Mande, N. Ramachandran, S. Salma Asiya Begum and F. Moreira (2024), even though this word does not use the ACO algorithm, but it uses the Coati algorithm, which belongs to the same class, swarm intelligence algorithms, the paper presents an Optimized Reinforcement Learning (ORL) approach for resource allocation in Vehicular Ad Hoc Networks (VANET). It combines Reinforcement Learning (RL) with Coati Optimization (ACO) to optimize channel access, reduce data collisions, and enhance data transmission efficiency. Implemented in MATLAB, the method outperforms existing techniques in terms of delay, packet delivery rate, throughput, and collision rate. The results demonstrate its potential for real-world VANET applications, addressing scalability, communication overhead, and hardware limitations [18].

From the works we discussed, Saeed Kaviani et al. (2021) [12], is the closest, as this study proposes a hybrid protocol that combines Q-learning and deep learning in a new protocol DeepCQ+ that mainly targets MANETs. But as we have pointed out earlier, scalability, realistic simulation, and addressing overheads are still challenges yet to be solved. In this paper, we introduce a novel hybrid approach that combines ACO with Q-learning, an approach that focuses on MANETs, dynamic parameter tuning with Q-Learning, and resource allocation optimization. As dynamically adjusting ACO parameters (like pheromone levels and evaporation rates) based on network performance. This leads to more adaptive and intelligent decision-making. While addressing scalability issues by using bigger numbers of nodes (50-80 nodes), with more realistic simulations that take into account heterogeneous MANETs, and providing a faster and a light-weight protocol to address the overhead issue.

3. Methodology and Implementation

3.1 Overview

The dynamic nature of MANETs requires adaptive algorithms for optimal performance. The ACO algorithm can struggle with the challenges such environments pose. To address this, we propose integrating ACO with Deep Q-learning (ACO-DQN) and traditional Q-learning (ACO-QL). Given the computational limitations of MANET nodes, ACO-QL is a more feasible alternative to ACO-DQN. Comparative analyses are conducted between ACO, and both ACO-DQN, and ACO-QL on a MANET simulation.

This section details the implementation of the three protocols, as well as the implementation of the MANET simulation. This study employs the NetworkX and Matplotlib libraries to implement and visualize network graphs, highlighting paths taken by each algorithm in distinct colors. To simulate the dynamic behavior of MANETs, the random waypoint model is used, where nodes move randomly within a confined area at a constant speed [19]. This simulation computes metrics including end-to-end delay, and algorithm runtime. The ACO-DQN is implemented using the PyTorch library. PyTorch is used to implement the deep learning component of ACO-DQN.

3.2 Implementation of a MANET Simulation

We intend for our simulation to be for research purposes only, straightforward, highly customizable, lightweight, easy to use as it is implemented in Python, and used for proof of concepts for researchers who want to manually implement their algorithms/protocols.

As mentioned in 3.1, the simulation is implemented using the NetworkX library, It includes functionality for initializing the network, updating node positions and topology, simulating energy consumption, node failures, visualizing the network, and calculating performance metrics. The node movement is simulated using an enhanced random waypoint model as nodes move randomly within the area and pause for a specified time, after movement, the graph topology (edges) are updated. Alongside, the random waypoint model to simulate mobile nodes, three parameters to simulate real-world mobility: 1) SPEED: Represents maximum device velocity (e.g., 2–5 m/s for pedestrians, 10–30 m/s for vehicles). 2) PAUSE_TIME: Mimics stationary intervals (e.g., 5–10 sec for temporary stops). 3) AREA_SIZE: Defines operational boundaries. 4) COMM_RANGE: Typical wireless range (e.g., 30–100 m for Wi-Fi/Bluetooth). 5) NUM_NODES: representing the number of nodes. The network is initialized where nodes are added with random positions and energy attributes. Edges between nodes are added if they are within communication range. The edges are initialized using a double loop with random weights but are updated using distance-based weights. A k-d tree is used to optimize edge updates and for efficient neighbor searches, replicating real-time link state changes in dynamic

environments and mirroring real-world link establishment/teardown due to mobility. The k-d tree is useful for range searches and nearest neighbors searches [20], edges are added or removed if nodes move in or out of range, and communication costs are represented by random weights for edges.

Energy management is done by calculating the energy consumed when traversing a path, as energy is reduced per hop, clamped to a minimum threshold, while node failures forcibly deplete energy to 0. Energy is deducted from nodes involved in the path, while ensuring energy doesn't drop below a specified minimum energy threshold. There are two parameters regarding energy consumption: 1) ENERGY_COST_PER_HOP: Energy consumed per transmission, scaled by distance (e.g., 0.5–2 Joules/hop). 2) MIN_ENERGY: Devices shut down below this threshold (e.g., 10% battery reserve).

As real MANETs could face node failure at multiple occasions, node failures are incorporated in our simulation based on a random failure probability, and energy is set to 0 for failed nodes marking them as inactive. Alongside energy depletion to simulate failure, a random failure parameter is incorporated, FAILURE_RATE, which simulates abrupt hardware/software faults (e.g., 1% failure rate per timestep).

The simulation provides functions to calculate the following metrics: 1) Path length: Sum of edge weights, and the number of hops. 2) End-to-end delay: Includes transmission delay (proportional to edge weight) and processing delay at intermediate nodes. 3) Packet delivery ratio. 4) Throughput. Execution time is calculated for each protocol too as another main metric, but it is implemented in each protocol implementation, and not as a part of the MANET simulation implementation.

However, when it came comes to limitations, this implementation has the following limitations:

1) This implementation omits environmental obstacles, interference, and signal reflection (e.g., urban multipath effects).
2) Does not account for congestion interface or MAC-layer collisions when it comes to packet loss
3) When it comes to energy depletion, it depends on the energy cost per hop in a linear function of distance, as the energy cost is calculated by multiplying the energy consumed per hop by the ratio of the distance to the communication range.
4) Does not model correlated failures (e.g., regional outages) or node recovery.

By taking these limitations into consideration, our MANET simulation implementation assumes the following:

1) By using the random waypoint model with constant speed and pause times, the simulation assumes that nodes move independently and randomly within a fixed area.
2) That all nodes are within a fixed communication range.
3) The packet loss model relies on a fixed loss probability combined with distance-based loss.
4) Ideal omnidirectional signal propagation.
5) Failures are simulated via random probability `(FAILURE_RATE)` and energy depletion.

As for visualization, it's done using the Matplotlib visualization library, where node sizes and colors represent energy levels, and paths are highlighted (e.g., discovered by algorithms like ACO-DQN), as shown in Fig. 1.

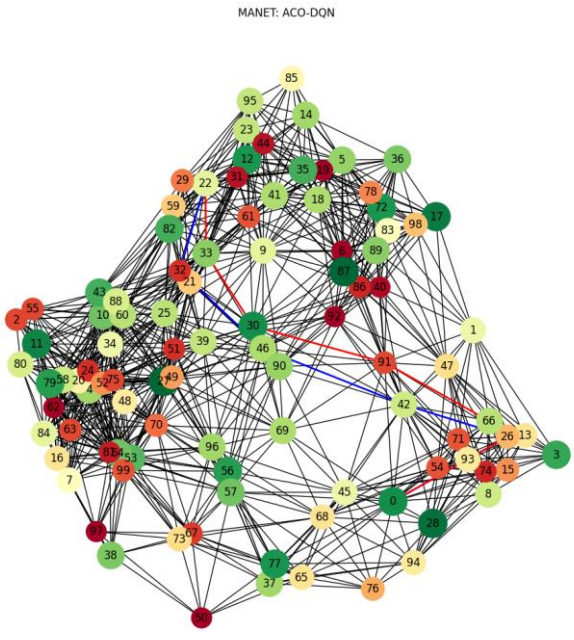

Fig. 1. A visualization of a network that compares between two different protocols: AntHocNet (blue path), and ACO-DQN (red path).

3.3 Implementation of AntHocNet

AntHocNet combines reactive path setup and proactive maintenance to efficiently route packets in MANETs. During reactive setup, forward ants explore paths by selecting next nodes using a hybrid probabilistic-deterministic rule: with probability q_0, the neighbor with the highest combined heuristic value is chosen (exploitation), otherwise neighbors are selected probabilistically (exploration).

```
if random() < q0:
        choose best neighbor (exploitation)
    else:
        choose probabilistically (exploration)
```

The heuristic integrates pheromone strength (τ_{ij}^{a}), distance inverse ($1/d_{ij}^{\beta}$), node energy weighting ($\left(E_j/E_{max}\right)^{\gamma}$), and stability (which prioritizes nodes with fewer neighbors to avoid congested nodes). Proactive maintenance periodically reinforces paths via backward ants, adjusts pheromones using energy-weighted rewards, and synchronizes pheromones with the current topology (e.g., removing stale nodes/edges), which is critical for handling node failures/mobility. Pheromones evaporate at rate ρ to encourage exploration, and bidirectional updates ensure routing symmetry. Initial pheromone levels are energy-aware, favoring high-energy nodes. Key parameters $(\alpha, \beta, \gamma, q_0, \rho)$ balance exploration, path quality, and adaptability to dynamic network conditions.

Overall, node selection is done as shown in equation (1):

$$p(i \to j) = \left(\frac{\tau_{ij}^{\alpha} \cdot \left(\left(\frac{1}{d_{ij}}\right)\right)^{\beta} \cdot \left(\left(\frac{E_j}{Emax}\right)\right)^{\gamma} \cdot S_j}{\Sigma_k \left[\tau_{ik}^{\alpha} \cdot \left(\left(\frac{1}{d_{ik}}\right)\right)^{\beta} \cdot \left(\left(\frac{E_k}{Emax}\right)\right)^{\gamma} \cdot S_k\right]} \right) \quad (1)$$

Where:

- τ_{ij}: Pheromone level
- d_{ij}: Edge distance/weight
- E_j: Neighbor energy
- S_j: Stability (1 / (1 + neighbor's degree))
- α, β, γ: Tuning weights.

The hyperparamters that can impact the protocol's performance are:

- q_0: the probability of choosing the bext neighbor, a high probality prioritizes exploitation, and vice versa.
- α: pheromone weight in node selection, high α over-relies on pheromones, causing path stagnation, a low one ignores historical data, reducing path stability.
- β: distance weight in node selection, the lower the value, the longer the paths, which balances energy and stability.
- ρ: pheromone evaporation rate, a high rate leads to rapidly forgetting paths, encouraging exploitation, a lower rate retains paths for longer, reducing adaptability to topology changes.
- γ: energy weight in heuristic, a higher value leads to strongly avoiding low-energy nodes.

3.4 Implementation of ACO-DQN

The ACO-DQN algorithm combines ACO with Deep Q-Networks (DQN) to find optimal paths in MANETs. It uses ant agents to explore paths while balancing exploration (random moves) and exploitation (learned Q-values) , pheromone updates are weighted by node energy levels to guide future exploration and reinforcement learning to train the neural network that predicts Q-values for node transitions.

There are three key components of this protocol: 1) The neural network (DQN), a fully connected architecture that takes normalized node features as an input, these features are x,y coordinates and energy level, and outputs Q-values for transitions between nodes. 2) Pheromone management, as phonemes are updated based on path quality and node energy, and evaporate over time. 3) Hybrid Selection Strategy: which balances between exploration (randomly select nodes with probability ε) and exploitation (choose nodes with highest combined Q-value and pheromone levels).

The algorithm starts by initializing the DQN with random weights, and setting initial pheromones τ_0 for all edges and exploration rate ε to 1.0. Each ant then starts at its respective source node and repeatedly selects next node until reaching the destination, selecting next nodes is done using the equation (2) after filtering them by checking if they energy more than the minimum energy threshold:

$$P(choose\ j) = \begin{cases} Random(j), & if\ \xi < \varepsilon\ (exploration) \\ argmax_j\ Q[(s_i, a_j).\tau_{ij}], & otherwise\ (exploitation) \end{cases} \quad (2)$$

Where:

- $Q(s_i, a_j)$: Predicted Q-value for transitioning from current state s_i (node i) to node j (action a_j).
- τ_{ij}: Pheromone level on the edge between nodes i and j.
- ξ: Random number between 0 and 1 (`ξ~Uniform(0,1)`).
- ε: Exploration rate (probability of random selection).

The next node is selected by randomly selecting one, which is the "exploitation" approach, or by selecting the one with highest score (i.e $Q(s_i, a_j) \times \tau_{ij}$) which benefits from learned Q-values and historical pheromone data to favor high-reward, energy-efficient paths. Then, the node is added to the path and marked as visited.

After selecting the next node, the pheromones are updated as shown in equation (3):

$$\tau_{ij} \leftarrow (1 - \rho).\tau_{ij} + \Sigma(reward\ .\ avg\_energy) \quad (3)$$

Where:

- ρ: Pheromone decay rate

As for Q-values, the DQN gets updated to improve its predictions of Q-values for node transitions as shown in equation (4), so it aligns its Q-value predictions with actual path performance (rewards), and balances immediate rewards (path cost) with future gains (discounted Q-values). Similarly, the DQN's updates adjust the DQN's weights to minimize this loss via backpropagation, the loss is calculated using mean squared error between predicted Q-values and target Q-values as shown in equation (5). Adjusting the DQN's weights is done by an Adam optimizer that predicts Q-values in the forward pass, calculates loss, and then updates the weights in the backward pass.

$$Q_{target} = reward + \gamma\ .\ max(Q(s_{next}, a_{next})) \quad (4)$$

Where:

- Reward: Inverse of the path cost (1 / (path cost + 1e-6)`), encouraging shorter/more efficient paths.
- γ: Discount factor, prioritizing immediate rewards over future ones.
- $\max(Q(s_{next}, a_{next}))$: Highest predicted Q-value for the next state (node).

$$loos = \left(Q_{target} - Q_{predicted}\right)^2 \quad (5)$$

Where:

- $Q_{predicted}$: DQN's current Q-value prediction for the action (node transition).

As, number of iterations increases, the exploration probability (ε) is reduced gradually to prioritize learned Q-values over time, so early iterations target high exploration to discover diverse paths, and later iterations target high exploitation to refine optimal paths using learned Q-values. This ensures the algorithm transitions from chaotic exploration to strategic exploitation. The decay in exploration probability (ε) is calculated as shown in equation (6):

$$\varepsilon = max\left(\varepsilon_{min}, \varepsilon, \varepsilon_{decay}\right) \quad (6)$$

Where:

- ε_{decay}: Decay rate (e.g., 0.95, ε reduces by 5% per iteration).
- ε_{min}: Minimum exploration probability (e.g., 0.01, ensures 1% random exploration).

The hyperparameters that impact this whole process are:

1) Number of ants for exploration.
2) ε_{decay}, which mangas the balance between exploration and exploitation.
3) ε_{min}.

4) Pheromone evaporation rate. High evaporation rate adapts quickly to topology changes, but forgets old paths faster. Low decay means longer memory of old paths but with slow adaption.
5) Learning rate, which is the step size for the Adam optimizer. A high learning rate is faster but can overshoot optimal solutions, and a low learning rate is slower but has a stable convergence.
6) Hidden layer sizes, which are the number of neurons in hidden layers, larger networks can model complex relationships but risk over fitting, and smaller networks train faster but risk underfiting.

Fine-tuning these parameters is automated using grid-search by using path cost as an evaluation metric to select the best set of values for the hyperparamters.

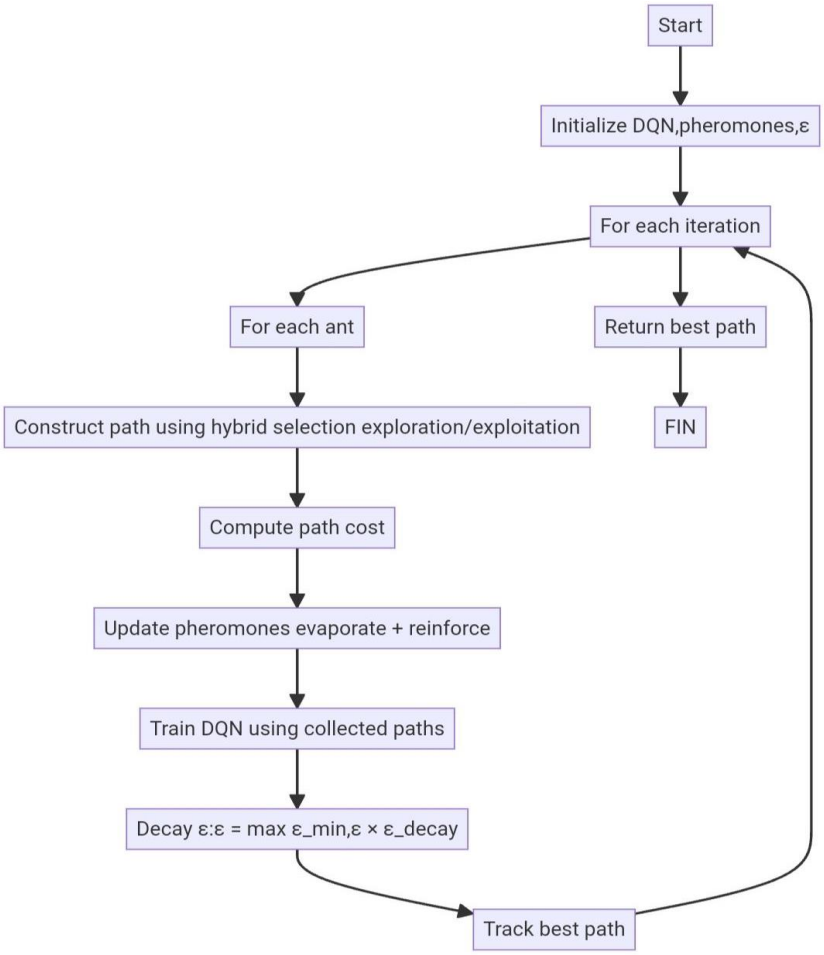

Fig. 2. ACO-DQN implementation flowchart

3.5 Implementation of ACO-QL

The implementation of the ACO-QL algorithm is simpler, it balances between exploration and exploitation when selecting neighbors, and it utilizes a table-based Q-Learning approach, where the Q-values for each node-action pair are explicitly stored in a table instead of maintaining a neural network and doing complex operations (e.g. back propagation). The table initializes values with energy penalties based on neighbor nodes' energy levels, as shown in equation (7). This prioritizes neighbors with higher energy and lower edge weights (e.g., shorter distances). So, in other words, the energy handling here needs to be more explicit unlike the implicit energy handling of ACO-DQN through node features. To ensure that nodes with low energy contribute less to Q-values updates, the rewards are modified based on the next node's energy, as shown in equation (8).

$$Q_{0(i,j)} = \left[\left(\frac{1}{1 + edge_{weight(i,j)}}\right)\right] . \left[1 - \left(1 - \left(\frac{E_j}{E_{max}}\right)\right)\right] \tag{7}$$

$$Q(i,j) \leftarrow Q(i,j) + \alpha . \left[\left(r . \left(\frac{E_j}{E(max)}\right)\right) + \gamma . \max(j,k) - Q(i,j)\right] \tag{8}$$

Where:

- γ: the discount factor.

Pheromone updates, again, directly influence action selection, and are updated using energy-weighted rewards. Selecting the next node works the same as in ACO-DQN, using an epsilon-greedy approach, but while ACO-DQN inherently models long-term rewards by adjusting the weights through backpropagation, a new hyperparameter was introduced in ACO-QL to weight future rewards, which is the discount factor γ. High discount factor prioritizes paths with higher long-term rewards, even if they have slightly higher immediate costs, as it encourages exploration of paths that may yield better long-term outcomes. . Low discount factor focuses on immediate rewards, favoring short-term gains, as it leads to faster exploitation of known good paths but risks missing optimal routes. The grid search for hyperparamters is not implemented here, to avoid the additional overhead it introduces.

Overall, the absence of the deep learning component makes ACO-QL a light-weight alternative for ACO-DQN, without the overhead of neural network training, grid search, and the complexity of the hidden layers as hyperparameters.

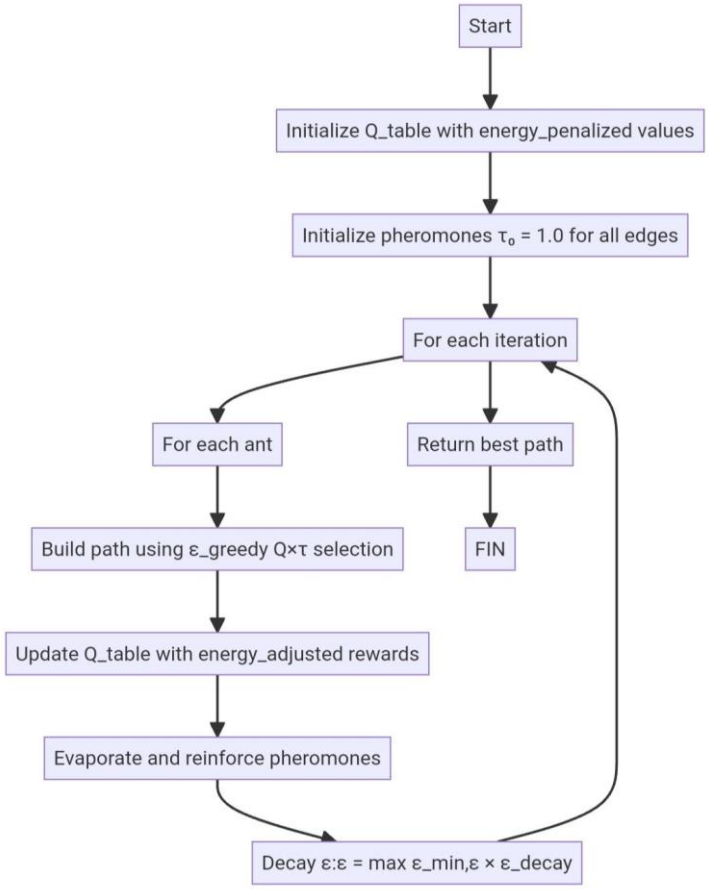

Fig. 3. ACO-QL implementation flowchart

4. Results

4.1 Overview

In this section, we present the results of comparing ACO-DQN protocol against the AntHocNet protocol and then the results of comparing ACO-QL protocol against AntHocNet protocol. The evaluation metrics for the comparisons are: path cost, end-to-end delay, and execution time. The protocols were evaluated using simulation under varying network configurations and iteration counts to assess their performance under different scenarios.

4.2 Experimental Setup

The objective of the experimentation is to evaluate the ACO-DQN (implemented using Pytorch) and ACO-QL protocols performances against the AntHocNet protocol on a MANET simulation, as the simulation is implemented using the NetworkX library and making the simulation as realistic as possible by incorporating all the traits of a real

MANET like energy consumption and node failure. The experiments are conducted using the T4 GPU setup in Google Colab.

The protocols are tested on three MANET configurations, the first configuration represents a stable network scenario. The second configuration represents an energy constrained scenario, where the minimum energy threshold and energy cost per hop are higher. The third scenario represents a high mobility node model with higher speed and lower pause time to test each protocol's robustness in a highly dynamic environment. The MANET settings are as shown in Table 1:

Table 1. The Three MANET Configuration

Config.	Comm Range	Area Size	# of Nodes	Energy Cost Per Hope	Max Energy	Min Energy	Speed	Pause Time	Failure Rate
Energy-constrained	30	100	80	1.0	100	20	2	5	0.01
Stable	30	100	50	0.5	100	10	2	5	0.01
High Mobility	30	100	50	0.5	100	10	3	4	0.05

As for the hyperparameters' settings, for the AntHocNet (Table 2), q_0 is the highest in the stable configuration because it allows exploiting known paths, unlike the other two less stable configurations, where exploration of new paths is encouraged as the topology changes frequently. Reliance on pheromones (α) is more needed in the stable configuration, as path stagnation is not an issue for a topology that does not change frequently. Longer paths are not encouraged in the second and third configurations as β lower than in the first configuration. Evaporation rate (ρ) is higher in the latter two scenarios to encourage fast adaptation in these more dynamic configurations. Energy weight γ in all three configurations is moderate. The shared hyperparameters between ACO-QL (Table 3) and AntHocNet are what vary considerably when it comes to ACO-QL, mainly, α, β, and ρ, as the same reasons for this difference apply here too. The number of ants was set to 50 for all three protocols, as it strikes a practical balance between exploration efficiency and network overhead using a fixed number ensures a fair and controlled comparison between protocols.

Table 2. AntHocNet's Hyperparameters' Settings

Hyperparamter	Stable MANET	Energy-constrained MANET	High Mobility MANET
q_0	0.7	0.3	0.3
α	2	1	1
β	1	2	2
ρ	0.2	0.6	0.6
γ	0.3	0.4	0.4

Table 3. ACO-QL's Hyperparamters' Settings

Hyperparamter	Stable MANET	Energy-constrained MANET	High Mobility MANET
Learning rate	0.15	0.1	0.1
Discount factor	0.75	0.6	0.6
ε	0.85	0.7	0.7
ε decay	0.96	0.95	0.95
ε minimum	0.7	0.1	0.1
Pheromone Decay	0.35	0.3	0.3

Tables 4-9 are examples of the experiments we conducted, with Tables 4-6 being comparisons between ACO-DQN and AntHocNet on three different MANET configurations, with Tables 7-9 being comparisons between ACO-QL and AntHocNet on two different MANET configurations.

4.3 Execution Time

We evaluate execution time by measuring the total runtime of each protocol for 30-75 iterations on a MANET configuration of 50-80 nodes. Execution time refers to the total time taken by the protocol to compute routes and run the protocol, and it is measured in seconds here.

Based on tables 4-9, we made the following observations:

1) ACO-DQN runs slower than AntHocNet in most cases, and it runs even slower when the number of iterations increases as shown in Table 4, it is important to point out that at lower iterations (10 or less), we found that ACO-DQN has similar or even faster execution times compared to AntHocNet.
2) ACO-QL mostly matches AntHocNet in terms of execution time regardless of the iterations count as shown in Tables 7-9.

We conclude that the ACO-DQN's deep learning overhead makes it inefficient at higher iterations. The ACO-QL protocol always runs faster than AntHocNet regardless of the iterations count, because there is no deep learning overhead.

4.4 Path Cost

We measure path cost as the cumulative distance of selected routes. Path cost quantifies the efficiency of the selected paths, with lower values indicating better optimization.

Based on Tables 4-9, we observed that in most experiments (e.g Table 6), AntHocNet and ACO-DQN have similar performances most of the time. While ACO-QL narrows the gap in this energy-constrained scenario (Tables 7-9).

We conclude that all three protocols remain competitive in terms of path cost.

4.5 End-to-End Delay

End-to-end delay measures the time taken for a packet to travel from source to destination. A lower delay indicates better performance [21].

In most of the experiments we conducted, AntHocNet consistently achieved lower delay than ACO-DQN and ACO-QL, but in rare cases such ACO-DQN achieved lower delay, but it does not justify the extreme deep learning overhead. ACO-QL consistently achieves lower delay than AntHocNet.

We conclude that ACO-QL's simplicity minimizes delays, making it ideal for latency-critical applications.

4.6 Throughput

We evaluate throughput as the average data rate (Mbps) sustained by the network, measuring routing efficiency under load [22]. Based on Tables 4-9, we made the following observations:

1) ACO-DQN outperforms AntHocNet in most cases, as shown in Tables 4-6.
2) ACO-QL outperforms AntHocNet in most cases, as shown in Tables 7-9.

We conclude that ACO-DQN's throughput superiority can justify its use in volatile environments, while ACO-QL/AntHocNet suffice for lightweight deployments.

4.7 Packet Delivery Ratio

We evaluate PDR as the percentage of successfully delivered packets, reflecting routing reliability under dynamic conditions [21]. Based on tables 4-9, we made the following observations:

1) ACO-DQN outperforms AntHocNet in most cases, as shown in Tables 4-6.
2) ACO-QL outperforms AntHocNet in most cases, as shown in Tables 7-9.

We conclude that ACO-DQN's learning capability maximizes reliability, which can justify its computational overhead in volatile environments. While ACO-QL/AntHocNet suffice for lightweight deployments.

Table 4. ACO-DQN vs. AntHocNet (Stable Environment)

Metric	ACO-DQN (Tuned)	AntHocNet
Best Cost	5	5
End-to-end Delay (s)	0.055	0.055
Time Taken (s)	41.36	0.273
PDR	88.9%	87.5%
Throughput (bps)	121378.13	119466.67

Table 5. ACO-DQN vs. AntHocNet (Energy-constarined Environment)

Metric	ACO-DQN (Tuned)	AntHocNet
Best Cost	4	4
End-to-end Delay (s)	0.045	0.045
Time Taken (s)	57.19	0.569
PDR	91.5%	70.3%
Throughput (bps)	124928.0	95982.93

Table 6. ACO-DQN vs AntHocNet (High Mobility Nodes)

Metric	ACO-DQN (Tuned)	AntHocNet
Best Cost	19	16
End-to-end Delay	0.205	0.205
Time Taken	37.88	0.220
PDR	70.3%	69.69%
Throughput (bps)	95982.933	95163.733

Table 7. ACO-QL vs. AntHocNet (Stable Environment)

Metric	ACO-QL	AntHocNet
Best Cost	7	10
End-to-end Delay (s)	0.08	0.14
Time Taken (s)	0.069	0.297
PDR	79.5%	41.9%
Throughput (bps)	108544.0	57207.47

Table 8. ACO-QL vs. AntHocNet (Energy-constarined Environment)

Metric	ACO-QL	AntHocNet
Best Cost	3	3
End-to-end Delay (s)	0.035	0.035
Time Taken (s)	0.067	0.843
PDR	89.2%	89.3%
Throughput (bps)	121787.73	121924.27

Table 9. ACO-QL vs AntHocNet (High Mobility Nodes))

Metric	ACO-QL	AntHocNet
Best Cost	11	11
End-to-end Delay	0.12	0.125
Time Taken	0.04366	0.2578
PDR	79.4%	72.3%
Throughput (bps)	108407.466	98713.6

5. Discussion

The AntHocNet protocol outperformed ACO-DQN in terms of path cost and end-to-end delay across most scenarios. However, ACO-DQN exhibited faster or similar execution time only under low iteration counts, as its simpler initial updates reduced computational overhead. At higher iterations, AntHocNet consistently achieved lower execution times, while ACO-DQN's runtime increased significantly due to its deep learning component. The ACO-QL protocol, in contrast, maintained a competitive execution times across all scenarios, owing to its lightweight Q-table updates.

These results suggest that AntHocNet's pheromone- and heuristic-driven approach excels at optimizing path quality and handling dynamic MANET traffic. ACO-DQN's initial speed advantage at low iterations diminished as the neural network's training overhead accumulated, aligning with its higher execution times. Despite extensive hyperparameter tuning (via grid search), ACO-DQN could not surpass AntHocNet's path cost or delay performance in energy-constrained or any other scenarios. This indicates that the deep learning layer in ACO-DQN may not fully leverage its potential in MANET routing or that the problem complexity does not justify its computational cost. However, ACO-DQN consistently outperformed AntHocNet when it comes to PDR and throughput, making more reliable in different scenarios, but it is important to point out that the gap between AntHocNet and ACO-DQN when it comes to PDR and throughput is rarely huge.

ACO-QL's efficiency stems from its lack of deep learning overhead, enabling faster execution times and lower delays than AntHocNet and ACO-DQN in all tested conditions. While ACO-QL matched AntHocNet's PDR and throughput in most cases. This trend in the results across all metrics continues even when we increased the number of nodes to 200 to address the scalability in a stable environment using the second MANET settings, as shown in Tables 10 and 11.

Table 10. ACO-DQN vs AntHocNet (200 Nodes)

Metric	ACO-DQN (Tuned)	AntHocNet
Best Cost	5	3
End-to-end Delay	0.06	0.06
Time Taken (s)	167.9	2.914
PDR	81.9%	69.4%
Throughput (bps)	111820.8	94754.13

Table 11. ACO-QL vs AntHocNet (200 Nodes)

Metric	ACO-QL	AntHocNet
Best Cost	4	4
End-to-end Delay	0.045	0.045
Time Taken (s)	0.363	2.147
PDR	89.3%	89.1%
Throughput (bps)	121924.27	121651.2

In conclusion, AntHocNet remains the most robust choice for dynamic, energy-aware MANETs, while ACO-QL offers a comparable and faster lightweight alternative for time-sensitive applications. ACO-DQN's performance advantages when it comes to PDR and throughput do not outweigh its computational costs in most practical scenarios.

6. Conclusion and Future Work

This study evaluated the performance of ACO-DQN and ACO-QL protocols against the established AntHocNet protocol in a MANET simulation. The results demonstrate that AntHocNet consistently outperforms both protocols in terms of path cost, and end-to-end delay, while ACO-DQN outperforms in terms of PDR and throughput. However, ACO-QL outperforms AntHocNet in terms of end-to-end delay, while mostly matching AntHocNet in other metrics, making it a viable lightweight alternative for scenarios where speed is important. The ACO-DQN adds unnecessary complexity, as the deep learning component does not improve the performance, and it introduces higher execution times at increased iterations.

Future work could explore hybrid approaches that integrate ACO-QL's Q-table efficiency into AntHocNet's proactive maintenance phase to accelerate route updates without sacrificing path optimization. Algorithmic improvements can be applied to ACO-DQN, like improving and speeding up the automated hyperparameters tuning. The MANET simulation can be enhanced by incorporating control overhead metrics (e.g., routing message frequency) to quantify scalability limits of proactive protocols like AntHocNet. ACO-QL can be deployed on IoT edge devices to benchmark real-time performance under hardware constraints.

References

[1] Khan, B. U. I., Olanrewaju, R. F., Anwar, F., Najeeb, A. R., & Yaacob, M. (n.d.). A Survey on MANETs: Architecture, Evolution, Applications, Security Issues and Solutions. Department of Electrical & Computer Engineering, Kulliyyah of Engineering, IIUM Malaysia. Retrieved Dec 28, 2024 from: https://www.researchgate.net/profile/Burhan-Khan-6/publication/327201905_A_Survey_on_MANETs_Architecture_Evolution_Applications_Security_Issues_and_Solutions/links/5b7fe70292851c1e122ec1a8/A-Survey-on-MANETs-Architecture-Evolution-Applications-Security-Issues-and-Solutions.pdf?__cf_chl_tk=Pr6L1X3eTs5QlqptyQl55poNAfqOqzkYW5M_Hs3IlCM-1735389667-1.0.1.1-uQAQHkvKs6I.APol5v8.yh3_VZ.Wv6_aWkGZLdH_yRc

[2] Goyal, P., Parmar, V., & Rishi, R. (n.d.). MANET: Vulnerabilities, Challenges, Attacks, Application. Department of Computer Science and Engineering, Technological Institute of Textile and Science, Bhiwani, Haryana, India. Retrieved Feb 1, 2025 from: https://www.researchgate.net/publication/289675441_Manet_Vulnerabilities_challenges_attacks_application

[3] M. Dorigo and L. M. Gambardella, "Ant colony system: a cooperative learning approach to the traveling salesman problem," in IEEE Transactions on Evolutionary Computation, vol. 1, no. 1, pp. 53-66, April 1997, doi: 10.1109/4235.585892. Retrieved Dec 28, 2024 from: https://ieeexplore.ieee.org/document/585892/authors

[4] Di Caro, G., Ducatelle, F. and Gambardella, L. M. (2004). AntHocNet: an Ant-Based Hybrid Routing Algorithm for Mobile Ad Hoc Networks. Proceedings of the 8th International Conference on Parallel Problem Solving from Nature (PPSN VIII). Retrieved Jan 10, 2025 from: https://people.idsia.ch/~frederick/anthocnet/anthocnet.html

[5] Uchhula, V., & Bhatt, B. (2010). Comparison of different Ant Colony Based Routing Algorithms. Dharamsinh Desai University, Nadiad, Gujarat, India. Retrieved March 28, 2025 from: https://www.researchgate.net/profile/Brijesh-Bhatt/publication/46122442_Comparison_of_different_Ant_Colony_Based_Routing_Algorithms/links/542bad320cf277d58e8a2281/Comparison-of-different-Ant-Colony-Based-Routing-Algorithms.pdf?origin=publication_detail&_tp=eyJjb250ZXh0Ijp7ImZpcnN0UGFnZSI6Il9kaXJlY3QiLCJwYWdlIjoicHVibGljYXRpb25Eb3dubG9hZCIsInByZXZpb3VzUGFnZSI6InB1YmxpY2F0aW9uIn19&__cf_chl_tk=0KKuhoOjrqibUl2SCaiisfDutUllODe4HwvlNaxwwSI-1743783981-1.0.1.1-UOwGQ4vwuqXHeY8wy_u3Ru3Gi7GvfC9y8vi3R2cWtAk

[6] Hao, Z.-F., Cai, R.-C., & Huang, H. (n.d.). An Adaptive Parameter Control Strategy for ACO. College of Computer Science and Engineering, South China University of Technology, Guangzhou, P. R. China; College of Mathematical Science, South China University of Technology, Guangzhou, P. R. China. Retrieved Feb 1, 2025 from: https://www.google.co.il/url?sa=t&source=web&rct=j&opi=89978449&url=https://www.researchgate.net/profile/Ruichu-Cai/publication/220776315_A_Novel_ACO_Algorithm_with_Adaptive_Parameter/links/0046352b24ede8caa2000000/A-Novel-ACO-Algorithm-with-Adaptive-Parameter.pdf&ved=2ahUKEwjtxK_c3KKLAxVPcKQEHb_aIaEQFnoECBEQAQ&usg=AOvVaw3le1goNLW0Wkgkqx5OYNUo

[7] S. Kaushik, K. Tripathi, R. Gupta and P. Mahajan, "Futuristic Analysis of Machine Learning Based Routing Protocols in Wireless Ad Hoc Networks," 2021 Fourth International Conference on Computational Intelligence and Communication Technologies (CCICT), Sonepat, India, 2021, pp. 324-329, doi: 10.1109/CCICT53244.2021.00067. Retrieved Nov 29, 2024 from: https://www.researchgate.net/publication/354110638_Futuristic_Analysis_of_Machine_Learning_Based_Routing_Protocols_in_Wireless_Ad_Hoc_Networks

[8] Arif, Mohammad & Bhargavi, K & Swaroopa, K & Karuturi, Satish & Balamurugan, A. (2024). Machine learning-Based Energy Efficient and Enhancing Communication Reliability for MANETs of Balanced Less Loss Routing Protocol. Journal of Electrical Systems. 20. 777-783. 10.52783/jes.1670. Retrieved Dec 3, 2024 from: https://www.proquest.com/openview/46e5efcf373e81cd1265b606b3c3ab04/1?pq-origsite=gscholar&cbl=4433095

[9] Akhilesh Bijalwan,Iqram Hussain, Kamlesh Chandra Purohit, and M. Anand Kumar. (2023). Enhanced Ant Colony Optimization for Vehicular Ad Hoc Networks Using Fittest Node Clustering. Retrieved Dec 19, 2024 from: https://www.mdpi.com/2071-1050/15/22/15903

[10] Zahid Khan, Sangsha Fang, Anis Koubaa, Pingzhi Fan, Fakhar Abbas, Haleem Farman, (2020) Street-centric routing scheme using ant colony optimization-based clustering for bus-based vehicular ad-hoc network, Computers & Electrical Engineering, Volume 86. Retrieved Dec 19, 2024 from: https://www.sciencedirect.com/science/article/abs/pii/S0045790620305917
[11] Z. Mammeri, "Reinforcement Learning Based Routing in Networks: Review and Classification of Approaches," in IEEE Access, vol. 7, pp. 55916-55950, 2019, doi: 10.1109/ACCESS.2019.2913776. Retrieved Nov 29, 2024 from: https://ieeexplore.ieee.org/abstract/document/8701570
[12] Saeed Kaviani, Bo Ryu, Ejaz Ahmed, Kevin A. Larson, Anh Le, Alex Yahja, Jae H. Kim. (2021). Robust and Scalable Routing with Multi-Agent Deep Reinforcement Learning for MANETs. Retrieved Dec 19, 2024 from: https://arxiv.org/abs/2101.03273
[13] H. -K. Hsin, E. -J. Chang, K. -Y. Su and A. -Y. Wu, "Ant Colony Optimization-Based Adaptive Network-on-Chip Routing Framework Using Network Information Region," in IEEE Transactions on Computers, vol. 64, no. 8, pp. 2119-2131, 1 Aug. 2015, doi: 10.1109/TC.2014.2366768. Retrieved Dec 19, 2024 from: https://ieeexplore.ieee.org/document/6945251
[14] R. Amin, E. Rojas, A. Aqdus, S. Ramzan, D. Casillas-Perez and J. M. Arco, "A Survey on Machine Learning Techniques for Routing Optimization in SDN," in IEEE Access, vol. 9, pp. 104582-104611, 2021, doi: 10.1109/ACCESS.2021.3099092. Retrieved Nov 29, 2024 from: https://ieeexplore.ieee.org/abstract/document/9493245
[15] Davi Ribeiro Militan, Hermes Pimenta de Moraes, Renata Lopes Rosa, Lunchakorn Wuttisittikulkij, Miguel Arjona Ram ́ırez, and Dem ́ostenes Zegarra Rodr ́iguez. (2021). Enhanced Routing Algorithm Based on Reinforcement Machine Learning—A Case of VoIP Service. Retrieved Nov 29, 2024 from: https://www.mdpi.com/1424-8220/21/2/504
[16] S. Nimmala, M. Ramchander, M. Mahendar, P. Manasa, D. D. Bhavani and K. Raghavendar, "Dynamic RL-ACO: Reinforcement Learning-based Ant Colony Optimization for Load Balancing in Cloud Networks," 2024 5th International Conference on Smart Electronics and Communication (ICOSEC), Trichy, India, 2024, pp. 475-480, doi: 10.1109/ICOSEC61587.2024.10722410. Retrieved Dec 19, 2024 from: https://ieeexplore.ieee.org/abstract/document/10722410
[17] Ghanshyam Prasad Dubey, Shalini Stalin, Omar Alqahtani, Areej Alasiry, Madhu Sharma, Aliya Aleryani, Piyush Kumar Shukla, M. Turki-Hadj Alouane, 2023, “Optimal path selection using reinforcement learning based ant colony optimization algorithm in IoT-Based wireless sensor networks with 5G technology”, Computer Communications, Volume 212. Retrieved Dec 19, 2024 from: https://www.sciencedirect.com/science/article/abs/pii/S0140366423003250
[18] S. Mande, N. Ramachandran, S. Salma Asiya Begum and F. Moreira, "Optimized Reinforcement Learning for Resource Allocation in Vehicular Ad Hoc Networks," in IEEE Access, vol. 12, pp. 167040-167048, 2024, doi: 10.1109/ACCESS.2024.3489395. Retrieved Dec 19, 2024 from: https://ieeexplore.ieee.org/abstract/document/10740275
[19] Johnson, David B., and David A. Maltz. “Dynamic Source Routing in Ad Hoc Wireless Networks.” Computer Science Department, Carnegie Mellon University, (1996). Retrieved Jan 10, 2025 from: https://link.springer.com/chapter/10.1007/978-0-585-29603-6_5#:~:text=This%20paper%20presents%20a%20protocol,which%20hosts%20move%20less%20frequently.
[20] Bentley, J. L. (1975). Multidimensional Binary Search Trees Used for Associative Searching. Communications of the ACM, 18(9), 509–517. https://doi.org/10.1145/361002.361007. Retrieved Jan 10, 2025 from https://dl.acm.org/doi/10.1145/361002.361007
[21] Kurose, J. F., & Ross, K. W. (2017).“Computer Networking: A Top-Down Approach” (7th ed.). Pearson. Retreived Jan 29, 2025 from: https://www.google.co.il/url?sa=t&source=web&rct=j&opi=89978449&url=https://www.ucg.ac.me/skladiste/blog_44233/objava_64433/fajlovi/Computer%2520Networking%2520_%2520A%2520Top%2520Down%2520Approach,%25207th,%2520converted.pdf&ved=2ahUKEwjC6ZCG5qKLAxW7TKQEHdE9Gz0QFnoECCUQAQ&usg=AOvVaw0nYJN2FwHGPeoye6BigjF5
[22] C. E. Perkins and E. M. Royer, "Ad-hoc on-demand distance vector routing," Proceedings WMCSA'99. Second IEEE Workshop on Mobile Computing Systems and Applications, New Orleans, LA, USA, 1999, pp. 90-100, doi: 10.1109/MCSA.1999.749281. Retrieved Mar 26, 2025 from: https://ieeexplore.ieee.org/document/749281

Authors’ Profiles

Yahia M. Abusaqer was born in Gaza, Palestine. He received a Bachelor's degree in computer science from the Islamic University of Gaza, Palestine, in 2022. He is currently pursuing a Master's degree in data science at the same university.He worked as a Teaching Assistant at the Islamic University of Gaza from August 2022 to October 2023, where he taught courses in assembly language, front-end development, back-end development, and freelancing online. His paper, "Exploring Arabic Sentiment Analysis: A Comparative Analysis of Techniques and Approaches," was accepted for oral presentation at the 11th International Conference on Advanced Technologies 2023, held in Istanbul, Turkey, from August 17 to August 19, 2023. His research interests include natural language processing, sentiment analysis, and applied machine learning.Mr. Abu Saqer was ranked second in his class during his undergraduate studies and is currently ranked first in his master's program.

Kahlil M. Eslayyeh was born in Gaza, Palestine. He received a Bachelor's degree in Information and Communication Technology (ICT) from the AL-Quds Open University of Gaza, Palestine, in 2016. He is currently studing for a Master's degree in Information Technology with specialization in Software Engineering at the Islamic University of Gaza, Palestine. He is currently working as an employee at the Ministry of Health with the job title Head of Nursing Information Technology Department since 2017 until now. He has contributed to supervising and developing several electronic applications in the healthcare sector. His research interests include Software Engineering, Information Security, and Machine Learning.

Nasser M. Abudalu was born in Gaza, Palestine. He is a skilled engineer with a Bachelor's degree in Multimedia Engineering from the University of Palestine in 2011. He has professional experience working with Palestine for Consulting and Marketing, as well as other companies such as Orange for internet services. His expertise includes graphic design, project management, Microsoft Office tools, customer service, strategic planning, and social media. Fluent in Arabic and English, he is well-versed in industry-standard software like Photoshop, PowerPoint, and Excel.

Aiman A. AbuSamra was born in Gaza, Palestine. He received a Ph.D. in computer systems from the National Technical University of Ukraine, Kiev Polytechnic Institute, Ukraine, in 1996. He is a Professor in the Computer Engineering Department at the Islamic University of Gaza, Palestine, where he has been working since 1996. His research focuses on computer networks, wireless sensor networks, network security, blockchain technology, and mobile computing. He has authored several publications, including "PalCert: A Blockchain-based Certificate Attestation and Verification System for HEIs in Palestine", a book chapter published in 2024, and journal articles such as "Enhancing the Lifetime of WSN Using a Modified Ant Colony Optimization Algorithm" (2023) and "Exploratory Study on Hyperledger Fabric Framework: Food Supply Chain as a Case Study" (2023). Prof. AbuSamra has contributed as a reviewer for the Journal of Information Security and Applications. His work has been recognized in various international conferences and journals, reflecting his expertise in network performance, security, and emerging technologies like blockchain.

How to cite this paper: Yahia Mohsen Abu Saqer, Khalil Mohammed Eslayyeh, Nasser Majed Abudalu, Aiman A. Abusamra, "ACO-QL: Enhancing ACO Algorithm for Routing in MANETs Using Reinforcement Learning", International Journal of Engineering and Manufacturing (IJEM), Vol.15, No.5, pp. 31-45, 2025. DOI:10.5815/ijem.2025.05.03