The Scientific Temper (2025) Vol. 16 (8): 4604-4616
E-ISSN: 2231-6396, ISSN: 0976-8653
Doi: 10.58414/SCIENTIFICTEMPER.2025.16.8.02
https://scientifictemper.com/

RESEARCH ARTICLE

AI-driven IoT routing: A hybrid deep reinforcement learning and shrike optimization framework for energy-efficient communication

V. Yasodha[1*], V. Sinthu Janita[2]

Abstract

The expansion of Internet of Things (IoT) networks has intensified the need for intelligent and adaptive routing strategies capable of handling frequent topological changes, energy limitations, and application-specific performance requirements. Existing routing protocols often struggle to simultaneously achieve scalability, energy conservation, and reliability. To address these challenges, this paper introduces a novel hybrid routing framework, DRL-SOA, which fuses deep reinforcement learning (DRL) with the shrike optimization algorithm (SOA) to enable real-time, congestion-aware, and energy-efficient routing in IoT environments. The DRL component incrementally learns optimal routing paths by interacting with dynamic network conditions, while SOA enhances the convergence of Q-learning by identifying the most promising action sequences using a nature-inspired hunting mechanism. The proposed method employs a multi-parameter fitness function that considers link stability, link duration, remaining energy, bandwidth availability, and node connectivity to determine optimal routing paths. Extensive simulations using NS-3 demonstrate that DRL-SOA significantly outperforms existing approaches, including RIATA, DRL-IRS, and DOACAR. Notably, the proposed approach achieves up to a 25% increase in network lifespan, reduces routing overhead by 22%, and enhances packet delivery and energy efficiency across different node densities and mobility rates. These results establish DRL-SOA as a scalable and robust routing protocol for next-generation IoT systems.

Keywords: Congestion-aware routing, Deep reinforcement learning, Energy efficiency, Internet of Things, Routing protocols, Shrike optimization algorithm.

关键词： 拥塞感知路由、深度强化学习、能源效率、物联网、路由协议、Shrike优化算法。

Introduction

The rapid expansion of the Internet of Things (IoT) has led to an unprecedented growth in the number of connected devices, requiring efficient, reliable, and energy-aware communication protocols. IoT networks often consist of numerous resource-constrained sensor nodes that operate in dynamic and heterogeneous environments. These networks face significant challenges such as limited energy resources, frequent topology changes due to node mobility, congestion, and the need for low-latency data transmission.

Traditional routing protocols designed for static or low-mobility wireless sensor networks are often inadequate for modern IoT applications, especially those requiring high mobility support (e.g., vehicular networks, wearable healthcare devices). Consequently, intelligent and adaptive routing strategies have gained increasing attention. Artificial intelligence (AI) and machine learning (ML) techniques, particularly reinforcement learning (RL) and meta-heuristic optimization algorithms, provide promising solutions by enabling nodes to learn optimal routing behaviors from environmental feedback.

Among AI/ML techniques, deep reinforcement learning (DRL) has emerged as a powerful tool for sequential decision-making in complex, dynamic systems. DRL combines reinforcement learning's trial-and-error paradigm with deep

[1]Assistant Professor, PG & Research Department of Computer Science, Cauvery College for Women (Autonomous), Affiliated to Bharathidasan University, Tiruchirappalli, India - 620 018.

[2]Head & Professor, PG & Research Department of Computer Science, Cauvery College for Women (Autonomous), Affiliated to Bharathidasan University, Tiruchirappalli, India - 620 018.

***Corresponding Author:** V. Yasodha, Assistant Professor, PG & Research Department of Computer Science, Cauvery College for Women (Autonomous), Affiliated to Bharathidasan University, Tiruchirappalli, India - 620 018, E-Mail: yasodha.ca@cauverycollege.ac.in

How to cite this article: Yasodha, V., Janita, V.S. (2025). AI-driven IoT routing: A hybrid deep reinforcement learning and shrike optimization framework for energy-efficient communication. The Scientific Temper, **16**(8):4604-4616.

Doi: 10.58414/SCIENTIFICTEMPER.2025.16.8.02

Source of support: Nil

Conflict of interest: None.

Received: 04/07/2025 **Accepted:** 21/07/2025 **Published:** 30/08/2025

neural networks' representation capability, allowing routing agents to learn optimal policies even in large state-action spaces. This capability is critical for IoT networks, where network states vary rapidly due to node mobility and energy depletion. DRL's adaptability helps achieve improved energy efficiency, congestion management, and route stability compared to conventional or heuristic approaches.

On the other hand, the Shrike Optimization Algorithm (SOA) is a population-based metaheuristic algorithm that is inspired by the natural behaviors of shrike birds, such as nesting, reproduction, and feeding. It simulates the lifecycle of shrikes to balance exploration and exploitation phases during optimization, making it well-suited for complex multi-objective problems such as routing in IoT networks. SOA's ability to quickly converge to near-optimal solutions while maintaining population diversity helps in selecting energy-efficient and congestion-free routes. Compared to other swarm intelligence algorithms (e.g., Particle Swarm Optimization, Ant Colony Optimization), SOA offers superior convergence speed and accuracy, with reduced computational complexity. In this work, SOA is applied to optimize Cluster Head (CH) and Relay Node (RLY) selection in Internet of Things (IoT) networks, focusing on energy efficiency, reliability, and adaptability.

The integration of DRL and SOA leverages the strengths of both techniques: DRL's adaptive policy learning and SOA's efficient optimization capabilities. This hybrid approach enables dynamic route selection that simultaneously maximizes network lifetime, minimizes delay, reduces congestion, and adapts to mobility. Unlike standalone DRL or heuristic methods, the hybrid algorithm balances long-term learning with fast convergence to near-optimal routes in real-time.

This paper proposes a novel Hybrid Deep Reinforcement Learning-based Shrike Optimization Algorithm (DRL-SOA) for energy-efficient, congestion-aware, and mobility-adaptive cluster-based routing in IoT networks. To the best of our knowledge, this is the first work combining DRL and SOA for IoT routing optimization. The protocol is evaluated against benchmark protocols including RIATA (mobility-aware routing), DRL-IRS (reinforcement learning-based routing), and our previously developed DOACAR protocol (Dingo Optimizer-based Congestion-Aware Routing), which demonstrated strong energy performance in static scenarios but lacked adaptability under high mobility.

The rest of this paper is organized as follows: Section II reviews related work on AI-based routing protocols. Section III details the proposed DRL-SOA protocol design and optimization objectives. Section IV presents simulation setup and performance evaluation. Finally, Section V concludes the paper with insights and future research directions.

Related works

Optimizing Cluster Head (CH) and Relay (RLY) node selection is essential to improve energy efficiency, scalability, and adaptability in Wireless Sensor Networks (WSNs) and Internet of Things (IoT) environments. Recent research has applied fuzzy logic, reinforcement learning (RL), and metaheuristic optimization techniques to enhance routing performance under dynamic and resource-constrained conditions.

Fuzzy logic-based methods such as the Type-2 fuzzy-based CH election scheme proposed by (Adnan, M., Ahmad, T., and Yang, T, 2021) have demonstrated improved energy efficiency and reliability. However, their practical deployment in large-scale networks is limited due to high computational complexity and poor scalability. Reinforcement learning has emerged as a promising technique for adaptive routing. (Bhimshetty, S., and Ikechukwu, A.V., 2024) employed Deep Q-Networks (DQN) for energy-aware routing in IoT, enabling real-time decision-making based on environmental conditions.

Nain, Z., Musaddiq, A., Qadri, Y.A., Nauman, A., Afzal, M.K., and Kim, S.W., (2021) introduced RIATA, a reinforcement learning-based routing protocol that uses trickle timers to reduce control overhead and improve Packet Delivery Ratio (PDR), though it does not address CH or RLY node selection.

Ahmad, S., Khan, S., Khan, K.S., Naeem, F., & Tariq, M. (2023) proposed a deep reinforcement learning approach for resource allocation in IRS-assisted networks. While it enhances packet delivery and offers adaptive learning, it primarily targets IRS scenarios rather than core CH/RLY optimization in WSNs. L. Amudavalli, and K. Muthuramalingam. (2024) proposed a location-based energy-efficient routing protocol using the Teaching–Learning Soccer League Optimization (TLSLO) algorithm for WSNs. Their work demonstrates the potential of metaheuristic optimization in improving clustering and routing performance, which inspires our DRL-SOA approach by emphasizing energy-aware CH selection and route formation.

Nimmala, S., Gupta, N.S., Sena, P.V., Chari, K.K., Pasha, M.A., & Rambabu, B. (2025) applied DQN for smart energy applications in WSNs, demonstrating strong energy adaptability and learning-based optimization, but requiring significant training time and coordination among nodes.

Metaheuristic algorithms are widely used due to their global search capabilities and simplicity. Rajoriya, M.K., & Gupta, C.P. (2023) implemented the Sailfish Optimization Algorithm (SFO) for CH selection in Software-Defined WSNs, achieving energy-aware multi-hop routing but facing limitations in highly dynamic topologies. Panimalar, S., & Jacob, T.P. (2024) developed a hybrid congestion-aware routing system that combines Bee Colony Optimization with Intelligent Butterfly Optimization to achieve load balancing. However, the real-time adaptability of this method is affected by increased system complexity.

Farag, H., & Stefanovič, Č. (2021) presented an RL-based congestion-aware routing protocol for dynamic IoT

networks. While their model adapts well to changing traffic conditions, it lacks dedicated mechanisms for CH and RLY node selection, which can reduce routing efficiency in dense deployments.

Among bio-inspired algorithms, the Shrike Optimization Algorithm (SHOA), proposed by AbdulKarim, H.K., & Rashid, T.A. (2024), is notable for its balance between exploration and exploitation, fast convergence, and low computational overhead. SHOA is lightweight and suitable for real-time WSN/IoT applications, though its real-world validation remains limited. Yasodha, V., & Janita, V.S. (2025) introduced DOACAR, a Dingo Optimizer-based protocol designed for congestion- and mobility-aware CH and RLY node selection. While DOACAR demonstrates enhanced PDR and energy efficiency, it lacks a learning mechanism to support dynamic topology changes in real-time.

To address these limitations, the proposed DRL-SOA framework integrates the learning ability of DRL [2, 3, 4, 5] with the optimization power of SOA by AbdulKarim, H.K., & Rashid, T.A. (2024). DRL enables context-aware, real-time routing decisions based on changing environmental conditions, while SOA ensures efficient CH and RLY node selection with minimal computational overhead. This hybrid approach enhances energy efficiency, adaptability, and congestion control, making DRL-SOA highly suitable for scalable and dynamic IoT environments. The advantages and limitations of the discussed methods are summarized in Table 1.

Proposed work

This research introduces DRL-SOA, a hybrid protocol combining Deep Reinforcement Learning (DRL) with the Shrike Optimization Algorithm (SOA) to enable energy-efficient, congestion-aware, and secure routing in IoT networks. The system comprises three key modules: DRL-based route learning, SOA-based cluster head (CH) and relay node (RLY) optimization, and a trust-aware congestion detection mechanism. The architecture ensures real-time adaptability, efficient packet forwarding, and enhanced network lifetime.

System Model

The dynamic and resource-constrained nature of Internet of Things (IoT) networks necessitates intelligent routing mechanisms that ensure efficient, reliable, and secure data transmission. To meet these demands, this work proposes a novel hybrid routing protocol, DRL-SOA, which integrates Deep Reinforcement Learning (DRL) and the Shrike Optimization Algorithm (SOA). The proposed protocol is designed to adapt to topological changes, optimize energy consumption, minimize congestion, and enhance routing trustworthiness.

The IoT network is modelled as a graph $G(N,C)$, where $N=\{n_1,n_2,n_3.....n_t\}$ denotes the set of sensor nodes, and C represents the set of communication links between node pairs (ni, nj) that lie within the coverage area A of the IoT environment. The transmission delay and the physical distance between nodes influence each network connection. Let S be the source node transmitting data to a destination node D through a set of intermediate nodes based on the link quality determined by inter-node distance and network conditions. This is illustrated in Figure 1.

To ensure robust performance, the protocol employs a Fitness Function (FF) that incorporates key network

Table 1: Comparative Summary of Recent Routing Protocols

Author & Year	*Method*	*Advantages*	*Limitations*
Adnan *et al*. (2021)	Type-2 Fuzzy Logic	Energy-efficient, reliable	High complexity, scalability issues
Bhimshetty & Agughasi (2023)	DQN-based Reinforcement Learning	Learns optimal paths dynamically	Long training time, high computation
Nain *et al*. (2021)	RIATA (RL + Trickle Timer)	Low control overhead, high PDR	No CH/RLY optimization
Ahmad *et al*. (2023)	DRL-IRS (Deep RL)	Improves packet delivery, adaptive learning	High training complexity and resource usage
Rajoriya & Gupta (2023)	Sailfish Optimization (SFO)	Energy-aware CH selection, efficient clustering	Limited in dynamic topologies
Farag & Stefanovič (2021)	RL-based Congestion-Aware Routing	Supports dynamic environments	Lacks CH/RLY selection, limited energy awareness
Panimalar & Jacob (2024)	Bee Colony + Intelligent Butterfly Optimization	Load balancing, congestion-aware routing	Complexity in dynamic adaptation
Nimmala *et al*. (2025)	DQN-based Routing for Smart Energy	Energy adaptability, learning-based routing	Requires coordination and training time
AbdulKarim & Rashid (2024)	SHOA (Shrike Optimization Algorithm)	Fast convergence, lightweight, global search	Limited real-world validation
Yasodha & Janita (2025)	DOACAR (Dingo Optimizer)	Congestion-aware, energy-efficient, scalable	Needs learning mechanism for dynamic networks

parameters such as Link Stability Factor (LSF), Link Duration Factor (LDF), Residual Energy, Available Bandwidth, Received Signal Strength Indicator (RSSI), and Network Connectivity. These parameters are used as weights in route selection to fulfill application-specific Quality of Service (QoS) requirements. Additionally, traffic is classified based on priority to facilitate critical data delivery, while a trust-aware mechanism is implemented to avoid unreliable nodes.

The DRL component enables the protocol to learn optimal routing policies in real time by interacting with the environment and updating decisions based on current network states. In parallel, the SOA module enhances the convergence of DRL by optimizing the selection of Cluster Heads (CHs) and Relay Nodes (RNs) using a bio-inspired strategy based on the shrike's hunting and feeding behavior. This combination ensures dynamic route optimization, improved network lifetime, reduced control overhead, and enhanced packet delivery.

Objective parameters evaluation

Link stability factor (LSF)

In the proposed approach, the Link Stability Factor (LSF) is employed to dynamically adjust the learning rate (alpha) in the Deep Reinforcement Learning model, thereby contributing to the formation of more stable routes during routing updates. This factor plays a pivotal role in making informed and adaptive routing decisions under dynamic network conditions.

Consider a communication link k=(a,b), where $k \in \mathcal{L}$, and L represents the set of all active links. At a given time' s_i', the positions of nodes a and b are denoted as $\left(x_a^{s_i}, y_a^{s_i}\right)$ and $\left(x_b^{s_i}, y_b^{s_i}\right)$, respectively. At a subsequent time $s_{i+1} = s_i + \Delta s$, the updated positions of the nodes are $\left(x_a^{s_{i+1}}, y_a^{s_{i+1}}\right)$ and $\left(x_a^{s_{i+1}}, y_b^{s_{i+1}}\right)$ as obtained through periodic beacon exchanges. The Euclidean distances between nodes a and b at times s_i and s_{i+1} are denoted as $DT^{s_i}(a,b)$ and $DT^{s_{i+1}}(a,b)$ respectively. These distance variations are used to compute the LSF, which reflects the temporal consistency of a link and guides the routing protocol in selecting stable communication paths.

$$DT^{s_i}(a,b) = \sqrt{\left(x_a^{s_i} - x_b^{s_i}\right)^2 + \left(y_a^{s_i} - y_b^{s_i}\right)^2} \quad (1)$$

For a given communication link $k = (a,b)$, the change in connectivity distance is defined as:

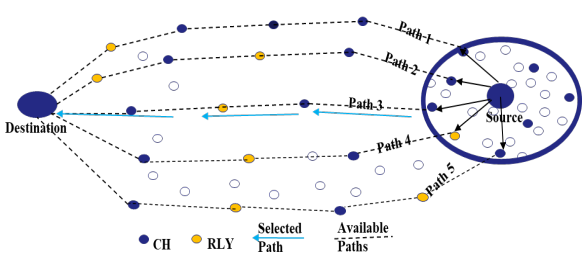

Figure 1: Data communication in IoT

$|\Delta CD(k)| = |DT^{s_{i+1}}(a,b) - DT^{s_i}(a,b)|$, where $DT^{s_i}(a,b)$ *and* $DT^{s_{i+1}}(a,b)$ represent the Euclidean distances between nodes a and b at times 's_i' and 's_{i+1}' respectively. This metric quantifies the variation in the physical distance between the nodes over the interval $\Delta s = s_{i+1} - s_i$. A smaller value of |ΔCD(k)| indicates higher link stability, implying minimal relative movement between the nodes. Conversely, a larger variation reflects high mobility and unstable connectivity.

The Link Stability Factor (LSF) for the link k=(a,b) over interval Δs is calculated as:

$$LF_{\Delta s}(k) = \frac{|\Delta CD(k)|}{\Delta CD_{max}} \quad (2)$$

Where,

$\Delta CD_{max} = 2 \cdot a_{max} \cdot \Delta s$ and 'a_{max}' denotes the maximum node speed. A lower LSF value corresponds to more consistent inter-node distance, signifying a more stable and long-lasting link. This stability allows nodes to remain within effective communication range for extended periods, thereby reducing the frequency of route recompilations and contributing to more efficient Q-learning updates with minimal adjustments.

Link Duration Factor

The **Link Duration Factor (LDF)** quantifies the predicted or observed time interval during which a communication link between two IoT devices remains intact before breaking. It is governed by the relative **speed** and **direction of motion** of the nodes involved. By evaluating LDF, routing protocols can assess the link's stability, enabling them to make proactive decisions for maintaining seamless connectivity. This minimizing the chances of packet loss during mobility-induced transitions and ensuring higher Quality of Service (QoS) in dynamic IoT environments.

The quantity of variation in speed between two IoT devices in the direction of 'v, p, q' is given by,

$$w_v = \bar{r}_i \sin\theta_i \cos\tau_i - \bar{r}_j \sin\theta_j \cos\tau_j \quad (3)$$

$$w_p = \bar{r}_i \sin\theta_i \sin\tau_i - \bar{r}_j \sin\theta_j \sin\tau_j \quad (4)$$

$$w_q = \bar{r}_i \cos\tau_i - \bar{r}_j \cos\tau_j \quad (5)$$

Let '$\bar{r}_i, \bar{r}_j$' represent the average velocities of nodes i and j, respectively. The directional variations between the two nodes are described in the 3D space by angles θi, θj (in the $v-p$ plane) and τi, τj (in the z-axis). These parameters capture node dynamics in spatial dimensions.

The duration of communication link 'T_{ij}' between nodes i and j is defined as the period during which the squared differences in their movement parameters (velocity and direction vectors) remain within a predefined maximum distance threshold d_{max}^2.

$$m = w_v^2 + w_p^2 + w_q^2 \quad (6)$$

$$n = 2\left[w_v \left(v_i - v_j \right) + w_p \left(p_i - p_j \right) + w_q \left(q_i - q_j \right) \right] \quad (7)$$

$$p = \left(v_i - v_j \right)^2 + \left(p_i - p_j \right)^2 + \left(q_i - q_j \right)^2 \quad (8)$$

By using the above derivations (6), (7) and (8), 'T_{ij}' is given by,

$$T_{ij} = \frac{-n \pm \sqrt{n^2 - 4mp}}{2m} \quad (9)$$

Where, $d_{max}^2 = \left(v_i - v_j + w_x T_{ij} \right)^2 + \left(p_i - p_j + w_p T_{ij} \right)^2 + \left(q_i - q_j + w_q T_{ij} \right)^2$

A larger LDF indicates a more stable link, thereby improving the reliability of the route over time. This prediction capability is essential for minimizing packet loss, enabling smooth handovers, and maintaining a high Quality of Service (QoS) in mobile IoT environments.

Available Bandwidth

The primary goal of routing is to find suitable paths to the destination, regardless of the current network load or application requirements. However, this can lead to network congestion when traffic is high, which reduces overall performance. To maintain efficiency, it is important to check the available bandwidth along the selected path. Throughput is limited by the lowest bandwidth node along the route, also known as the bottleneck. Therefore, selecting paths with higher available bandwidth ensures better data transfer. Measuring residual bandwidth using the IEEE 802.11 MAC protocol is not straightforward. To overcome this, the study uses a method from earlier research that calculates the Channel Occupancy Factor (COF), which indicates how much of the wireless channel is being used by mobile nodes in the area.

Each node monitors radio activity continuously to estimate the available bandwidth locally. This is done over a fixed observation period, called (T_{BW} /sec), which is set to 1 second. During this time, each node records how long it uses the channel. To detect whether the channel is idle or busy, two sensing techniques are used. Physical sensing includes energy detection and carrier sensing, while virtual sensing relies on the Network Allocation Vector (NAV). The channel is considered idle when the node is neither transmitting nor receiving, and the wait time exceeds the NAV value. It is marked as busy if the node is active or if the wait time is less than or equal to the NAV. Based on these observations, the COF is calculated using Equation (10), helping determine how much bandwidth is still available for use.

$$COF(v) = \frac{Cl_{busy}(T_{BW})}{T_{BW}} \quad (10)$$

The channel's busy state is calculated using Eq. (11) by using Figure 2. where t_{BC_i} represents the time the channel is busy due to each node i.

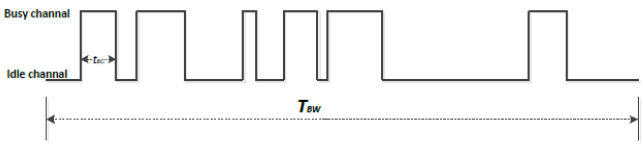

Fig. 2: Channel Busy State

$$l_{busy}(T_{BW}) = \sum_i t_{BC_i} \quad (11)$$

A windowed mean approach is employed to precisely determine the COF. Hosts determine the available bandwidth for fresh data transfers by multiplying channel's total bandwidth by proportion of idle time within a specific timeframe. Available bandwidth can be concisely represented by Eq. (12).

$$AV_{BW}(v) = \frac{AV_{BW}}{Cl_{BW}} = 1 - COF(v) \quad (12)$$

Nodes Residual Energy (RE)

Residual energy (RE) of a node plays a dominant role in mobile IoT networks that showing the lifecycle of the network and equilibrium of energy consumption. To comprehend this equilibrium in the network, it is essential to firmly regulate the nodes with more RE to contribute to forwarding task. $\frac{Ey_t(i)}{Ey_{init}(i) - Ey_t(i)}$

The factor '$\frac{Ey_t(i)}{Ey_{init}(i) - Ey_t(i)}$' is included in the objective function, wherein '$Ey_{init}(i)$' and '$Ey_t(i)$' are initial and RE of node (i) correspondingly. Concurrently, factor '$\overline{Ey_i(TP_i)} = \frac{1}{NN} \sum_{j=1}^{NN} \frac{Ey_t(j)}{Ey_{init}(j) - Ey_t(j)}$' is included to mean RE of adjacent node.

NN - Amount of adjacent nodes of 'i' for 'TP_i' TP_i - Transmitting Power

Network Connectivity

Ensuring network connectivity is vital for regular network functioning. By including this factor, it can be ensured that the network remains connected after several game iterations, when transmitting power at a node is reduced. Connection function is given by,

$$NC(x_i, x_j) = \begin{cases} 1, & Link\ connected \\ 0, & Else \end{cases} \quad (13)$$

Objective function

Using the derived parameters for LSF, link duration quality, energy consumption, bandwidth availability, RSSI, and network connectivity, we form an objective function as follows:

$$U = \alpha_1 * \left(\frac{Ey_t(i)}{Ey_{init}(i) - Ey_t(i)} \right) + \alpha_2 \times T_{ij}(n) + \alpha_3 \times \left(LF_{\Delta s}(k) \right) + \alpha_4 \times \left(NC_{(x_i, x_j)}(n) \right) + \alpha_5 \times ABWF(v) \quad (14)$$

Where,

$á_1, á_2, á_3, á_4$ and $á_5$ - Weight factors

The objective function of proposed work is,

$$OBJ_{i,j} = e^{-U} \quad (15)$$

The objective function is limited to the range [0, 1]. Four main criteria are considered when determining the objective function for improving QoS of network. To reduce number of retransmissions, node coverage and distance amid nodes are considered, and to stabilize node energy, energy metric is taken into consideration, while the success rate improves the trustworthiness of data delivery.

Hybrid deep reinforcement learning and Shrike Optimization Algorithm (DRL-SOA)

DRL background

Our complex routing problem aligns with decentralized partially observable markov decision process (Dec-POMDP) framework, commonly employed for collaborative decision-making among multiple agents (Bhimshetty, S., & Ikechukwu, A.V., 2024). Dec-POMDPs model and optimize agent behavior in uncertain environments by considering the actions $\mathcal{A}$, observations $\mathcal{O}$, and states $\mathcal{S}$ of all involved agents. A POMDP is characterized by a set of potential system configurations (states), a range of possible actions, and a collection of observable outcomes. Actions are selected probabilistically based on a policy $i_a : \mathcal{O} \times \mathcal{A} \to [0,1]$ (given by δ), influencing the system's transition to a new state governed by the transition function of the environment $\mathcal{T} : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$. This transition yields a reward based on present state as well as action $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, as defined by the reward function. Agents perceive the environment through observations $o : \mathcal{S} \to \mathcal{O}$, which are functions of the system state. An overview of DRL algorithms and schemes used for developing hybrid routing policies for the problem is provided in the ensuing sections.

Value Optimization and Deep Q-learning

To determine the ideal policy, Q-learning, an extensively used model-free algorithm, is employed to estimate action-value (AV) function for a given policy $(V^i(s,a))$. AV function (Q-function) represents predictable cumulative discounted reward obtained by implementing actions 'a' in state (s) and consequently following optimal policy. Mathematically, it is defined as:

$$V^{\mu}(s,a) := \mathbb{E}[\sum_{t=0}^{T} \lambda^t r_t \mid s_t = s, a_t = a] \quad (16)$$

Where, 'ë' indicates discount factor and 'T' indicates time horizon The AV function is computed recursively as shown below.

$$V^{\mu}(s,a) = \mathbb{E}_{s'}\left[r(s,a) + \lambda \mathbb{E}_{a' \sim \mu}[V^{\mu}(s',a')]\right] \quad (17)$$

Recent DL models have enabled RL algorithms to approximate the Q-function using Deep Neural Network (DNN), i.e. $V(s,a) \approx V(s,a;\delta)$, where 'ä' represents a collection of NN parameters (i.e. objective function parameters) Deep Q-Network (DQN), as employed by Nimmala, S., Gupta, N.S., Sena, P.V., Chari, K.K., Pasha, M.A., & Rambabu, B. (2025), is used to learn the optimal AV function (V^*) by reducing the prediction loss.

Recent advancements in deep learning have empowered reinforcement learning algorithms to approximate Q-function using DNNs. This function represented as $V(s,a) \approx V(s,a;\delta)$ is parameterized by NN weights 'δ' (LS, link duration, available bandwidth etc.). A prominent technique in this domain is Deep Q-Networks (DQN) [5]. DQN learns ideal AV function by reducing the following loss:

$$g(\delta) = \mathbb{E}_{(s,a,r,s')}\left[\left(V^*(s,a;\delta) - \left(r + \lambda \max_{a'} \bar{V}^*(s',a')\right)\right)^2\right] \quad (18)$$

Where, '$\bar{V}$' is a target network with parameters which are synchronized periodically with primary network's parameters (δ) to enhance training stability. DQN further employs a large experience replay buffer to store past experiences (s,a,r,s'). Ultimately, the ideal deterministic policy is determined by selecting action with maximum Q-value for the given state as shown below and derived once, optimal parameters δ^* are determined.

$$\mu^*(s) = \arg\max_{a \in \mathcal{A}} V(s,a;\delta^*) \quad (19)$$

Policy Optimizations

Policy gradient schemes offer an alternative approach to directly optimize the policy parameters without explicitly estimating the expected return. A stochastic policy $\mu(a_t \mid s_t)$ assigns probabilities to actions (a_t) for state (s_t) parameterized by 'δ. The objective in policy optimization is to increase the expected discounted return function by adjusting these policy parameters.

$$J(\delta) = \mathbb{E}_{s_0,a_0,s_1,\ldots}\left[\underbrace{\sum_{t=0}^{T} \lambda^t r_t}_{R_\tau}\right] \quad (20)$$

An experience sequence (or trajectory), denoted as 'τ' is composed of states, actions and rewards as $\{s_0, a_0, s_1, \ldots\}$ with $s_0 \sim p_0(s_0)$ and $\mu a_t \sim s(a_t \mid s_t)$, $s_{t+1} \sim \mathcal{T}(s_{t+1} \mid s_t, a_t)$. The discounted return is 'R_δ' for sequence 'τ', where 'p_0' is initial state distribution. The AV function (V_μ), state-value function (V_μ) as well as advantage function (A_μ) are given by:

$$V_\mu(s_t, a_t) = \mathbb{E}_{s_{t+1},a_{t+1},\ldots}\left[\sum_{l=0}^{T} \lambda^l r_{t+l}\right] \quad (21)$$

$$F_\mu(s_t) = \mathbb{E}_{a_t,s_{t+1},\ldots}\left[\sum_{l=0}^{T} \lambda^l r_{t+l}\right] \quad (22)$$

$$A_\mu(s,a) = V_\mu(s,a) - F_\mu(s) \quad (23)$$

Where,

$$a_t \sim \mu(a_t \mid s_t) \text{ and } s_{t+1} \sim \mathcal{T}(s_{t+1} \mid s_t, a_t)$$

Shrike Optimization Algorithm (SOA)

The Shrike Optimization Algorithm (SOA) is a population-based optimization technique inspired by the natural behaviors of shrike birds, including nesting, reproduction, and survival. In SOA, the population consists of multiple nests, each containing two dominant parent birds and a set of offspring called nestlings. These nests represent potential solutions to optimization problems, where the best solution within a nest is termed the local best, and the best across all nests is identified as the global best. In this research, SOA is applied to address a multi-objective optimization problem involving the selection of Cluster Head (CH) and Relay (RLY) nodes in Internet of Things (IoT) networks (Figure 3). Each bird or candidate solution represents a possible CH/RLY configuration, evaluated using parameters such as residual energy, Received Signal Strength Indicator (RSSI), delay, link stability, and node connectivity. The local best solution reflects the learned action based on Q-values, while the global best represents the optimal strategy for routing decisions. By simulating the shrike's adaptive and goal-directed behavior, SOA efficiently explores and exploits the solution space to ensure energy-efficient and reliable communication in dynamic IoT environments.

SOA begins by initializing key parameters: the population size (N), the number of nestlings per nest (B), and a natural influence factor (C). The algorithm starts with a population of N nests, each containing two randomly generated parent birds. Once the initial population is established, B nestlings are generated for each nest. The population can be represented by Eq. (24).

$$\text{Population}(N) = \begin{pmatrix} \begin{bmatrix} p_{im} & p_{if} \\ n_{ij} & n_{ij} \end{bmatrix} & \cdots & \begin{bmatrix} p_{im} & p_{if} \\ n_{ij} & n_{ij} \end{bmatrix} \\ \vdots & \ddots & \vdots \\ \begin{bmatrix} p_{im} & p_{if} \\ n_{ij} & n_{ij} \end{bmatrix} & \cdots & \begin{bmatrix} p_{im} & p_{if} \\ n_{ij} & n_{ij} \end{bmatrix} \end{pmatrix} \quad (24)$$

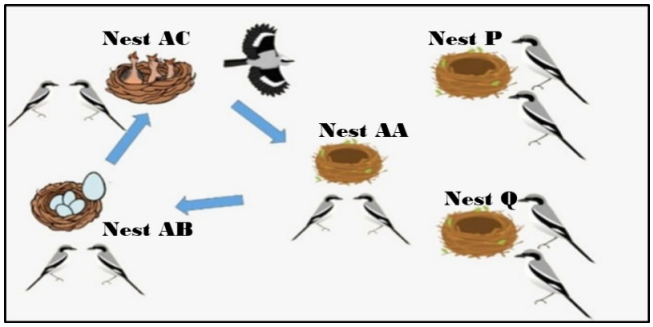

Fig. 3: Shrike Bird Life Cycle

Consider a population modeled as a pool of N nests, where each nest $\text{nest}_i\ (For\ i = 1\ to\ N)$ represents a candidate solution space. Each nest contains two parent solutions and a set of nestlings, all of which contribute to the optimization process. The parent solutions are initialized randomly within the defined search boundaries using Eq. (25). Here, LB and UB denote the lower and upper bounds of the solution space, respectively, and rand is a uniformly distributed random number

$$p_i = LB + \text{rand}(UB - LB) \quad (25)$$

During the initialization process, two birds are created as parents for each nest. The most physically fit bird will then be selected as the dominant male (M_{parent}) while the remaining bird becomes the dominant female (F_{parent}). In the breeding phase, each nest will produce a total of B nestlings using Eq. (26) and Eq. (27). The value of 'Δegg_j' is generated from both parents, and a random value (r) is chosen from the range of -1 to 1. This value is then used to generate the ' $stling_j$', where 'i' ranges from 1 to B.

$$\Delta\ egg_j = (F_{parent} - M_{parent}) + r \quad (26)$$

$$nestling_j = F_{parent} + \Delta\ egg_j \quad (27)$$

The young birds rely heavily on their parents for food, with the male parent taking on a dominant role in feeding. However, the male only feeds independently, while the female is capable of both independent feeding and feeding the nestlings if necessary. This system of dominant parental feeding leads to the search for and exploitation of optimal solutions. Each nest has two dominant parents, with the first being considered the optimal solution and the second being the backup. During the exploit phase, each nestling will be fed by their parent, leading to the convergence to the optimum solution. In the SOA algorithm, after initializing the nests and defining the parameters for the parents, the value for each dimension will be determined using Eq. (28), based on their specific objective function.

$$r = e^{-2xt/T_{max}} \quad \text{-------} \quad (28)$$

The 'r' parameter plays a critical role in feeding and is calculated to enhance exploration. The dimension variable for the '$bird_j$' is denoted by 'x', while 't' stands for the current iteration, and 'T_{max}' represents the maximum allowed iteration for performing SOA. Utilizing Eq. (29), every parent bird will be able to feed itself.w

$$\Delta\ food_j = bird_j \times r \quad \text{-------(29)}$$

However, when it comes to nourishing their hatchlings, the value of Δ food is calculated differently according to Eq. (30).

In this case, 'bird_j' represents the current state of the bird, and 'M_{parent}' is the male parent responsible for providing food.

$$\Delta\,\text{food}_j = r \times \left(\text{bird}_j - M_{parent} \right) + M_{parent} \tag{30}$$

Even though the nestlings were not able to survive solely on food provided by the male parent, they attempted to survive by relying on the female parent and using Eq. (31), which is equivalent to Eq. (30) but with r ranging from $-1\,\text{to}1$, and incorporating sin(α), here 'α' is a constant factor.

$$\Delta\,\text{food}_j = r \times \left(\text{bird}_j - F_{parent} \right) + \sin(\alpha) \tag{31}$$

Once the birds receive food, their positions are updated to reflect the new state using the following equation,

$$\text{bird}_j^{t+1} = \text{bird}_j^{t} + \Delta\text{food}_j \tag{32}$$

This update ensures that each bird's behavior adapts based on its feeding outcome. The fitness of each bird is evaluated, and the one with the highest fitness, denoted as 'bird_j^{t+1}', is selected to proceed to the next generation. However, not all birds may successfully receive food in every iteration. If a bird ('bird_j') does not receive nourishment from its parent, it still attempts to adapt by generating a new food value using a randomized exploration mechanism.

This is achieved by using Eq. (32) to generate a new amount of food 'Δfood_j'. This equation considers a randomly generated value, r, between the range of -1 to 1, as well as a variable parameter (α). This 'α' is generated randomly between 0 and the dimension of the problem, and is used to increase the element of randomness in the process. By utilizing the sine of this variable, the values will vary over time and lead to different solutions being explored. Incorporating sin(α) introduces diversity into the search, allowing the bird to explore solutions further away from its current state.

This strategy is critical for escaping local optima and enhancing the diversity of the search space. By randomly searching and diverging from the local best solution, the algorithm can generate new, potentially more optimal solutions. The corresponding update equation is given by Eq. (33)

$$\text{bird}_j^{t+1} = \text{bird}_j^{t} + \left(r \times \text{bird}_j + \sin(\alpha) \right) \tag{33}$$

The SOA employs a unique strategy to achieve optimal solutions. It maintains a record of the best solution found at each nest as its local best. The overall population then selects the best solutions from all local best to serve as the global best. This approach effectively tackles the issue of multiple modes by using a group of solutions. Each nest contains several birds (solutions), and after a fixed number of iterations 'k', old birds are replaced by new ones. However, the two best birds in each nest are kept as parents to guide the next generation. This helps the population gradually move toward better solutions.

SOA is chosen for its excellent balance between exploration and exploitation, mimicking shrike hunting behaviour. It offers low computational overhead, making it ideal for resource-constrained IoT nodes. The algorithm achieves fast convergence with high accuracy, suitable for real-time CH/RLY updates. It inherently supports multi-objective optimization across energy, RSSI, delay, and connectivity. Unlike other algorithms, SHOA remains robust and adaptive in dynamic and mobile IoT environments.

DRL Algorithm Based on SOA for Optimal Path Selection

This section presents DRL-SOA, a hybrid routing algorithm that integrates Deep Q-Learning with the Shrike Optimization Algorithm (SOA) to achieve efficient path selection in IoT networks. In this framework, SOA supports the reinforcement learning process by improving the exploration of possible routes, leading to more effective Q-value updates. By combining learning-based decision-making with bio-inspired optimization, the proposed model enhances adaptability, reduces energy consumption, and improves routing performance in dynamic and resource-constrained IoT environments.

Deep Reinforcement Q-Learning Framework Based on SOA

DRL-SOA's architecture combines the strengths of deep Q-learning and SOA to enhance overall performance of the model. The use of SOA allows for dynamic exploration and exploitation of the search space, while the deep Q-learning component helps to make more informed decisions by utilizing previously learned information. In this architecture, each feature or state is represented by a nest, and each participant (nest or node) is considered a population in SOA. This allows for a parallel and collaborative optimization process, further enhancing the model's performance. The Q-values in every nest represent the learning actions, which are constantly updated based on the fitness values determined by the environment. This allows for the model to adapt and learn in a dynamic and changing environment. Additionally, the incorporation of SOA allows for the identification of the local and global best sequences of learning actions, which further improves the overall performance of the model. Through a fixed number of iterations, DRL-SOA considers a new set of nests each time, providing a continuous and dynamic learning process. This process of updating and optimizing the learning actions in a collaborative and parallel manner leads to a more efficient and effective reinforcement learning model. As a result, DRL-SOA can successfully handle complex and highly dynamic environments, making it a powerful tool for applications in fields such as robotics, gaming, and finance.

Updating of Q-function based on SOA

This paper presents a novel approach for updating Q-function by adopting the SOA algorithm. The focus is on improving the overall performance of IoT networks through optimized route selection. The problem is considered as an optimization problem, where the fitness value plays a crucial role. To define the problem, the paper introduces two sets, $\mathcal{S}=\{s_1,s_2,\ldots..\}$ and $\mathcal{A}=\{\acute{a}_1,\acute{a}_2,\ldots\}$, representing states ($s_i$) and learning actions ($\alpha_i$) respectively. The reward value (r_t) is computed based on objective function of node by considering the present state (s_t) and learning action ($\acute{a}_t$). State, action and goal state are defined (Ahmad, S., Khan, S., Khan, K.S., Naeem, F., & Tariq, M., 2023). As the system moves from one state to another, the reward at a given time is determined by the state transition function, which is defined as $\mathcal{S}\times\mathcal{A}\rightarrow\mathcal{S}\times\mathbb{R}$ with $\mathcal{T}(s_t,\alpha_t)=(\bar{s_{t+1}},r_t)$. In this function, the system moves from state $\bar{s_{t+1}}$, obtaining a reward (r_t) at time (t). The aim is to identify ideal set of learning actions $\mathcal{A}=\{\acute{a}_t,\acute{a}_{t+1},\ldots.\}$ that improves total reward (R) based on FF $\mathcal{T}_{s_t}(\mathcal{A})=R(s_t,\mathcal{A})$ at time 't'.

The total reward is the total rewards of the user, where $r_t=\sum_{i=1}^{n}r_t^i$. At time 't', the learning agent obtains a reward (r_t) when performing a learning action (α_t) in state (s_t). The paper proposes DRL-SOA's algorithm which combines deep Q-learning with shrike optimization to find optimal set of actions in state i.e., nest (s). The objective is to maximize total reward over an extended period of time leading to better overall performance. Thus, the paper defines FF as total reward for Q-values and its calculated as follows.

$$\mathcal{T}_{s_t}(\mathcal{A})=R(s_t,\mathcal{A})=\sum_{t=1}^{T}\lambda^{T-t}r_t \tag{34}$$

With $\mathcal{T}(s_t,\alpha_t)=(\bar{s_{t+1}},r_t)$ in one episode of learning, total amount of learning actions is represented by 'T'. The present reward for 't^{th}' learning action is denoted by 'r_t', while the discount factor which has a value between 0 and 1 is represented by '$\ddot{e}$'.

The IoT optimal route selection problem is solved by finding ideal set of learning actions denoted by $=\{\acute{a}_t,\acute{a}_{t+1},\ldots.\}$ based on their corresponding Q-values for the local best. The global best is signified by state transition probability of selecting a specific learning action (α) with regard to 's' which lies in the range [0, 1]. To update solutions, a propounded approach is utilized by comparing local best solution got by every learning agent and global best solution got by every learning agent by using a FF. Fitness is calculated for nodes, and best target Q-value represented by local best '$bird_i^{t+1}$' is found by respective 'i^{th}' learning agent. The global best is determined by combining all the local best solutions, representing the best target Q-value for learning agents based on fitness of each node on route. The Q-function for each learning agent is updated using both local as well as global best target Q-values as defined by proposed strategy is given by,

$$bird_{j+1}^{(s_t,\alpha_t)}=bird_j^{(s_t,\alpha_t)}+\left(r\times bird_j+\sin(\alpha)\right) \tag{35}$$

$$Gbird_{j+1}^{(s_t,\alpha_t)}=\sum_{j=1}^{n}bird_{j+1}^{(s_t,\alpha_t)} \tag{36}$$

The variables 'r' and 'α' are used to represent random numbers with values between -1 and 1, while the variable 'α' is used to represent a random function that has a range of 0 to the dimension. The variable '$bird_j^{(s_t,\alpha_t)}$' signifies the Q-value of the 'i^{th}' nest when it chooses to take action (α_t) in 's_t' at 't'. The term '$bird_{j+1}^{(s_t,\alpha_t)}$' signifies local best Q-value found by 'i^{th}' nest or learning agent. Global best Q-value is obtained by considering all local best Q-values and using them in Eq. (35).

Proposed DRL-SOA algorithm

DRL-SOA is a routing method that combines deep reinforcement Q-learning with the Shrike Optimization Algorithm. As described in Algorithm 1, it starts by initializing the Q-learning process with a value table '$V(s,\alpha)$' and a random factor 'r', which simulates natural behaviour. The *r* value, ranging from –1 to 1, represents variation in environmental patterns like feeding. The learning agent begins in an initial state (s_1). At each time step (t), it takes an action (α_t), moves to a new state (s_{t+1}), and receives a reward (r_t). This experience is saved in memory as a tuple ($e_t=\langle s_t,\ \alpha_t,\ r,\ \bar{s_{t+1}}\rangle$) for learning. For each user, DRL-SOA performs three main steps are, (i) It calculates the cumulative reward (R) using Eq. (33), (ii) It identifies the best local and global Q-values based on the chosen actions (A), (iii) It updates these Q-values using Eq. (35). This cycle repeats over multiple iterations to find the best routes for each user in the network.

Algorithm 1: Psudocode for Deep Reinforcement Q-Learning Using Shrike Optimization

Input: Reward (R) - Cumulative reward for '$V(s,\alpha)$',
Local best reward (R^L) - Cumulative reward for '$bird_{j+1}^{(s_t,\alpha_t)}$'
Global best reward (R^G) - Cumulative reward for $Gbird(s,\alpha)=\sum_{j=1}^{n}bird_{j+1}^{(s_t,\alpha_t)}$
Max_it - Maximum amount of iterations
Set replay buffer size
Set '$V_j(s,\ \alpha)$' and current state '$bird_j^{V(s,\alpha)}$'
for (every user)
for ($j=1$ to Max_it)
for (every bird $i=1,\ldots,m$)
Initialize 's'
for ($t=1,2,\ldots,\tau$)
Execute learning action 'α_t'
Determine next state '$\bar{s_{t+1}}$' and reward 'r_t'
Store $e_t=\langle s_t,\ \alpha_t,\ r,\ \bar{s_{t+1}}\rangle$ in buffer

Compute cumulative reward R_i using Eq. (34)

// Update local best target

if ($R_i < R_i^L$ or $R_i^L == -1$) then

$bird_{j+1}^{(s_t,\alpha_t)} = V_i(s_t, \alpha_t)$

$R_i^L = R_i$

end if

// Determine Nest's global best target value

if ($R_i < R^G$ or $R^G == -1$) then

$G(s_t, \alpha_t) = V_i(st, \alpha t)$

$R^G = R_i$

end if

end for

Update local best target value '$bird_{j+1}^{(s_t,\alpha_t)}$' using Eq. (34)

Update global best target value '$G_{j+1}^{(s,\alpha)}$' by using Eq. (35)

end for

end for

end for

As shown in Algorithm 1, key parameters for the SOA such as the number of nests N, maximum iterations $Max_{Iteration}$, constant B, learning rate α, and exploration factor k are first initialized. During each iteration, the cumulative reward (or objective function) for each nest is calculated using Eq. (34). If the current fitness is greater than local best target (based on Q-values), it is assigned as the new local best Q-value. Otherwise, it remains the same. Among all nests, the one with the highest fitness becomes the global best. The target positions of nests are updated using Equations (35) and (36), guided by both local and global best Q-values. The algorithm continues until the maximum number of iterations is reached.

Results and Discussions

The proposed DRL-SOA (Hybrid Deep Reinforcement Learning with Shrike Optimization) was tested using the NS-3.23 network simulator to check its performance in changing IoT environments. The simulation settings, shown in Table 2, were chosen to reflect real-world IoT setups with dense and mobile nodes. The main performance measures were network lifetime, delay, energy use, packet delivery ratio (PDR), routing overhead, and throughput.

DRL-SOA was compared with three well-known IoT routing protocols: RIATA, DRL-IRS, and DOACAR. RIATA handles mobility but has high overhead and poor scalability. DRL-IRS saves energy using reinforcement learning but is computationally heavy. DOACAR works well in stable networks but struggles in dynamic ones. Overall, DRL-SOA showed better performance by using energy efficiently and making smart routing choices, even in mobile and large-scale networks.

Impact of Node Density on Protocol Performance

Network lifetime

DRL-SOA improves network lifetime by 40–60% compared to RIATA and DRL-IRS by using energy- and bandwidth-aware route selection. This prevents overuse of certain nodes and spreads energy usage evenly. Unlike RIATA and DOACAR, which lack adaptive energy control, DRL-SOA supports longer operation in dense networks.

End-to-end delay

DRL-SOA achieved the lowest end-to-end delay of 0.65 ms at 50 nodes, significantly outperforming RIATA, DRL-IRS, and DOACAR. Its real-time learning quickly finds optimal paths and avoids frequent retransmissions. This ensures fast and stable communication, ideal for time-sensitive IoT applications (Figure 4).

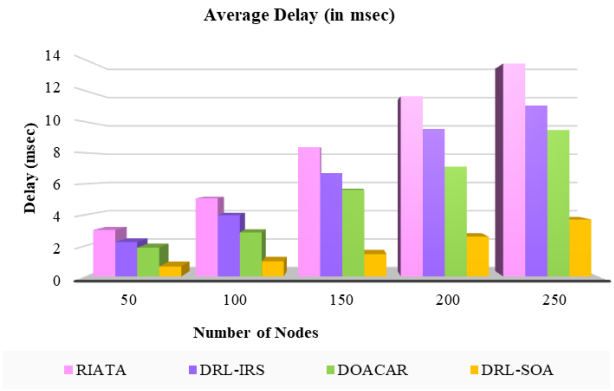

Figure 4: Average Delay (in msec)

Table 2: Presents the simulation setup details

Parameter	*Value*	*Parameter*	*Value*
Simulator	NS-3.23	Number of nodes	50 to 250
Topology	Random Node placement	Packet size	512 bytes
Mobility Model	Random Waypoint (RWP)	Packet rate	2Kb/sec
Speed	10 m/s to 50 m/s	Routing Algorithm	DRL-SOA
Pause Time	10 seconds	Initial energy	100J
Dimensions	1500*1500 m	Simulation Time	200Sec

Energy consumption

DRL-SOA consumed just 2.34 Joules at 250 nodes, saving up to 78.5% energy compared to RIATA (10.9 J). Its hybrid design reduces control overhead and avoids frequent route rediscovery. By factoring in energy and transmission cost, it ensures more efficient, longer-lasting communication (Figure 5).

Packet delivery ratio (PDR)

DRL-SOA achieves over 98% packet delivery ratio even at 50 m/s mobility, outperforming RIATA (85%) and DOACAR (90%) with a 13% gain over RIATA. Its predictive learning and dynamic path adaptation help maintain reliable delivery. This makes it highly effective in mobile and frequently changing IoT networks (Figure 6).

Routing overhead

DRL-SOA records just 10.26 bytes/sec routing overhead at 250 nodes—60% lower than RIATA (25.56), 52% lower than DRL-IRS (21.3), and 44% lower than DOACAR (18.4). This efficiency comes from minimizing redundant route updates and control packets. In contrast, other protocols rely on frequent network broadcasts, increasing overhead under mobility (Figure 7).

Throughput

DRL-SOA maintains over 45 Kbps throughput at high mobility, outperforming RIATA which falls below 35 Kbps (30% improvement). This is due to its stable path selection

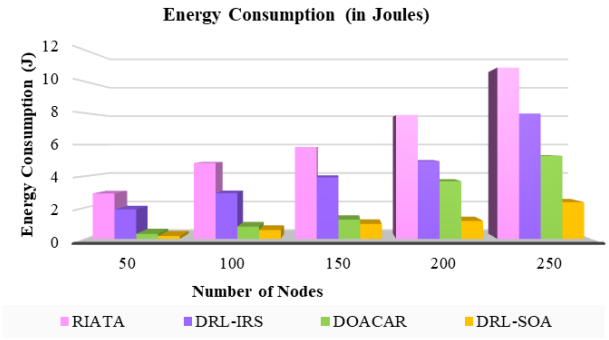

Figure 5: Energy Consumption (in Joules)

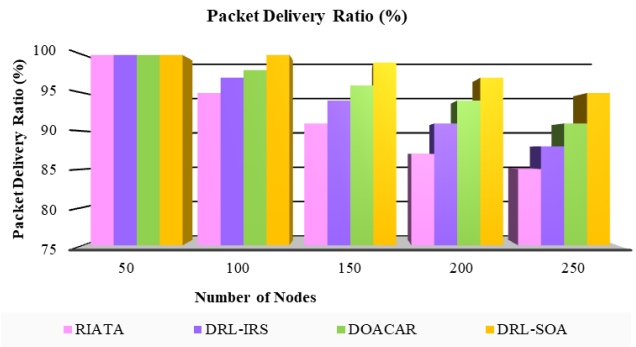

Figure 6: Packet Delivery Ratio (%)

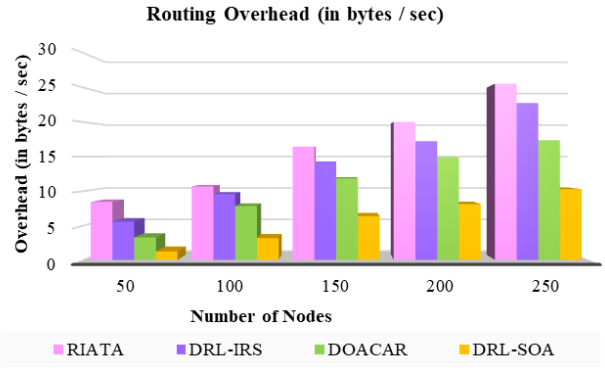

Figure 7: Routing Overhead (in bytes / sec)

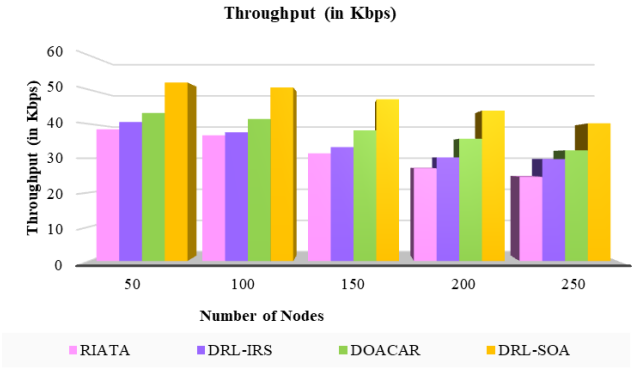

Figure 8: Throughput (in Kbps)

and lower packet loss from congestion or link failures. Its learning-optimization hybrid ensures reliable, high-speed transmission even in dynamic IoT conditions (Figure 8).

Overall, DRL-SOA demonstrated superior performance across all evaluated metrics, outperforming RIATA, DRL-IRS, and DOACAR. The proposed protocol improved network lifetime by up to 60%, reduced energy consumption by 78.5%, lowered delay by 78%, enhanced PDR by 13%, decreased routing overhead by 60%, and increased throughput by 30%. These improvements validate the effectiveness of integrating deep reinforcement learning with the Shrike Optimization Algorithm to deliver a scalable, energy-efficient, and reliable routing solution for dynamic IoT environments.

Impact of Node Speed on Protocol Performance

In mobile IoT systems, node mobility is a critical factor in real-time environments, such as smart healthcare and vehicle networks. When devices move, the network changes quickly, impacting routing. To test how well DRL-SOA handles this, node speeds were adjusted from 10 m/s to 50 m/s to reflect real-world movement. The Random Waypoint Mobility Model was employed, allowing nodes to move in random directions, pause, and then move again. This setup creates frequent changes in the network. By testing different speeds, we assessed how well DRL-SOA adapts, manages delays, and

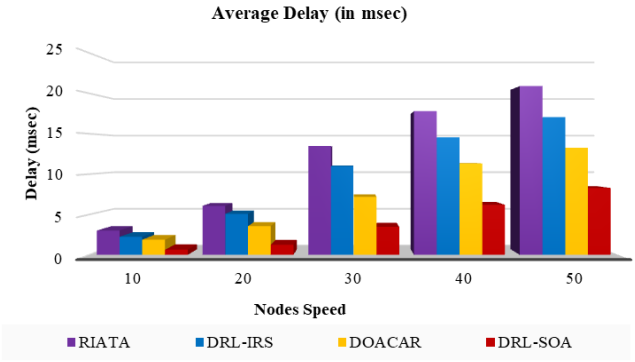

Figure 9: Average Delay (in msec)

remains reliable as movement increases. The results revealed significant changes in performance as speed increased.

End-to-end delay

As node speed increased, all protocols showed higher delays due to longer route setup and queuing. DRL-SOA maintained the lowest delay at 50 m/s (8.28 ms), outperforming RIATA (20.91 ms), DRL-IRS (16.64 ms), and DOACAR (13.47 ms). Its learning-based prediction helps avoid disruptions, reducing delay by up to 60% (Figure 9).

Energy Consumption

DRL-SOA consumes 2.85 J at 50m/s, significantly lower than RIATA (15.32 J), DRL-IRS (11.24 J), and DOACAR (7.93 J), and an energy saving of 81%, 74.6%, and 64%, respectively. This efficiency comes from DRL-SOA's adaptive routing, which considers residual energy and bandwidth. Its hybrid learning strategy reduces unnecessary transmissions, conserving energy even in high-mobility environments (Figure 10).

Packet Delivery Ratio (PDR)

As node speed increased, PDR dropped for all protocols due to more route breaks and unstable links. DRL-SOA maintained the highest PDR at 97%, outperforming RIATA (83%), DRL-IRS (87%), and DOACAR (87%), showing a 16.8% gain over RIATA. This reliability is due to DRL-SOA's adaptive

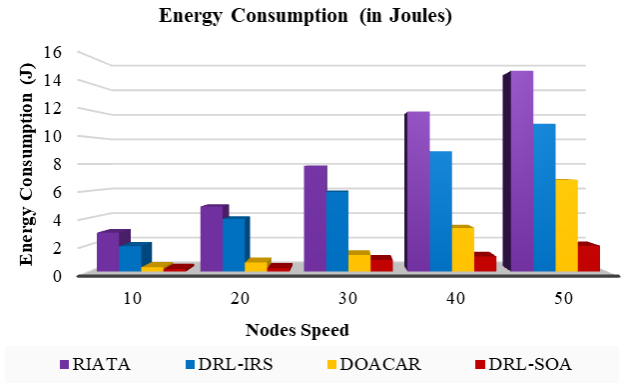

Figure 10: Energy Consumption (in Joules)

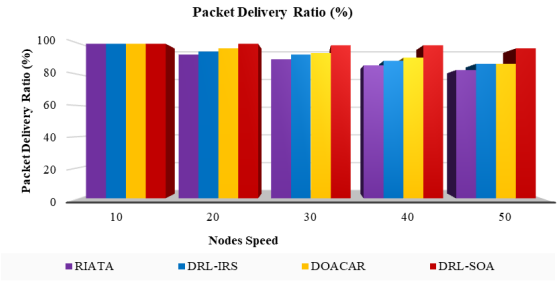

Figure 11: Packet Delivery Ratio (%)

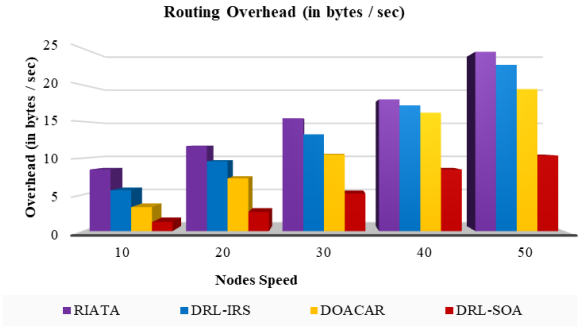

Figure 12: Routing Overhead (in bytes / sec)

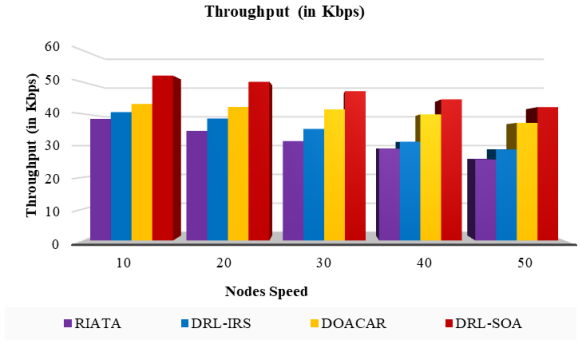

Figure 13: Throughput

learning and bandwidth-aware routing, which helps predict link failures and maintain stable paths (Figure 11).

Routing Overhead

As node speed increased, routing overhead rose for all protocols due to frequent route rediscovery and updates. DRL-SOA demonstrated the lowest overhead at 10.35 bytes/sec at 50 m/s, compared to 24.56 bytes/sec for RIATA, representing a 57.8% reduction. This efficiency results from DRL-SOA's selective control message exchange and policy-based updates, which minimize unnecessary routing traffic (Figure 12).

Throughput

Throughput dropped for all protocols as mobility increased due to more packet loss and unstable routes. However, DRL-SOA maintained the highest throughput at all speeds, reaching 42.16 Kbps at 50 m/s. This outperforms RIATA (25.56 Kbps), DRL-IRS (33.45 Kbps), and DOACAR (37.12 Kbps),

showing a 64.9% improvement over RIATA. This success is due to DRL-SOA's fast learning, efficient path selection, and reliable data transmission (Figure 13).

Varying node speed simulations offer valuable insights into routing performance under dynamic conditions. DRL-SOA consistently delivers stability, energy efficiency, and timely communication across mobility levels. This resilience highlights its suitability for mission-critical and latency-sensitive IoT applications.

Conclusion

This paper presents and evaluates the DRL-SOA protocol, which combines Deep Reinforcement Learning with the Shrike Optimization Algorithm. Its performance was tested against RIATA, DRL-IRS, and DOACAR in dynamic IoT environments. Key metrics such as packet loss, packet delivery ratio (PDR), routing overhead, throughput, average delay, and energy consumption were analyzed under different node speeds and densities.DRL-SOA consistently delivered better results than the compared protocols. It achieved lower packet loss, higher PDR, reduced routing overhead, better throughput, and shorter delays. Its intelligent learning and energy-aware routing approach helped maintain strong performance even in high-mobility scenarios.

The protocol also showed significant improvements in energy efficiency and scalability, making it well-suited for real-time, resource-limited IoT applications like smart healthcare, vehicular networks, and industrial automation.

These findings highlight the potential of combining deep learning with nature-inspired optimization techniques to tackle the challenges of modern wireless communication. In the future, the work will explore interference-aware routing to improve reliability in dense networks. There are also plans to develop hybrid optimization models and extend the protocol to support multi-hop communication and 6G-based IoT systems.

References

AbdulKarim, H. K., & Rashid, T. A. (2024). In search of excellence: SHOA as a competitive shrike optimization algorithm for multimodal problems. *IEEE Access*.

Adnan, M., Ahmad, T., & Yang, T. (2021, April). Type-2 fuzzy logic based energy-efficient cluster head election for multi-hop wireless sensor networks. In *2021 IEEE Asia Pacific Conference on Wireless and Mobile (APWiMob)* (pp. 32-38). IEEE.

Ahmad, S., Khan, S., Khan, K. S., Naeem, F., & Tariq, M. (2023). Resource allocation for IRS-assisted networks: A deep reinforcement learning approach. *IEEE Communications Standards Magazine*, *7*(3), 48-55.

Bhimshetty, S., & Ikechukwu, A. V. (2024). Energy-efficient deep Q-network: reinforcement learning for efficient routing protocol in wireless internet of things. *Indonesian Journal of Electrical Engineering and Computer Science*, *33*(2), 971-980.

Farag, H., & Stefanovič, Č. (2021, December). Congestion-aware routing in dynamic iot networks: A reinforcement learning approach. In *2021 IEEE Global Communications Conference (GLOBECOM)* (pp. 1-6). IEEE.

L. Amudavalli, & K. Muthuramalingam. (2024). Energy-efficient location-based routing protocol for wireless sensor networks using teaching-learning soccer league optimization (TLSLO). The Scientific Temper, 15(spl-1),32–44. https://doi.org/10.58414/SCIENTIFICTEMPER.2024.15.spl.05

Nain, Z., Musaddiq, A., Qadri, Y. A., Nauman, A., Afzal, M. K., & Kim, S. W. (2021). RIATA: A reinforcement learning-based intelligent routing update scheme for future generation IoT networks. *IEEE Access*, *9*, 81161-81172.

Nimmala, S., Gupta, N. S., Sena, P. V., Chari, K. K., Pasha, M. A., & Rambabu, B. (2025, April). Energy-Efficient Wireless Sensor Networks Optimization using Deep Q-Networks for Smart Energy Applications. In *2025 5th International Conference on Trends in Material Science and Inventive Materials (ICTMIM)* (pp. 1023-1027). IEEE.

Panimalar, S., & Jacob, T. P. (2024). A Congestion-Aware Routing System in Wireless Sensor Networks Based on Bee Colonies and Intelligent Butterfly Optimisation. *Wireless Personal Communications*, 1-18.

Rajoriya, M. K., & Gupta, C. P. (2023). Sailfish optimization-based controller selection (SFO-CS) for energy-aware multi-hop routing in software defined wireless sensor network (SDWSN). *International Journal of Information Technology*, *15*(7), 3935-3948.

Yasodha, V., & Janita, V. S. (2025). Optimizing IoT Communication: Context-Aware Energy and Congestion-Aware Cluster Routing. *International Journal of Intelligent Engineering & Systems*, *18*(6).