Computers, Materials & Continua

Tech Science Press

https://doi.org/10.32604/cmc.2026.084851

ARTICLE

Feasibility-Aware Reinforcement Learning for Reliable Hop-Constrained Routing in Wireless Sensor Networks

Adeel Iqbal[1,*,#] **and Muhammad Faisal Siddiqui**[2,*,#]

[1]School of Computer Science and Engineering, Yeungnam University, Gyeongsan-si, 38541, Republic of Korea
[2]Department of Computer Engineering, College of Computer Science and Information Technology, King Faisal University, Al Ahsa, 31982, Saudi Arabia

*Corresponding Authors: Muhammad Faisal Siddiqui. Email: msiddiqui@kfu.edu.sa; Adeel Iqbal. Email: adeeliqbal@yu.ac.kr

[#]These authors contributed equally to this work

Received: 30 April 2026; Accepted: 20 July 2026; Published: 13 August 2026

ABSTRACT: Hop-constrained packet routing is a fundamental problem in wireless sensor networks (WSNs), where latency constraints, energy limitations, and practical feasibility requirements greatly restrict routing choices. Traditional methods based on shortest path and greedy routing have low complexity but cannot adapt to dynamic network changes well, while reinforcement learning for routing has the potential to adapt to network variations but has not been well explored in the hard hop-constrained setting. The current study attempts to fill the gap by modeling hop-constrained routing as the decision-making problem in a finite-horizon setting. An integrated simulation environment is proposed that unifies the concept of feasibility-aware action masking, energy- and trust-aware routing mechanisms, and simulation-related evaluation criteria. In this unified environment, four representative reinforcement learning methods, REINFORCE, Advantage Actor–Critic (A2C), Proximal Policy Optimization (PPO), and Deep Q-Network (DQN), are applied and validated against greedy forwarding, shortest-path routing, and Dijkstra routing under strict ($H = 5$) and relaxed ($H = 15$) hop limits using multi-seed testing. Under tight hop constraints, Dijkstra achieves a delivery success rate of 1.000, while greedy routing reaches 0.950 ± 0.014. Among the learning algorithms, PPO, A2C, and DQN reach approximately 0.945 ± 0.014 at their best checkpoints with near-baseline hop efficiency, indicated by an average hop count of about 4.34 ± 0.04. Under relaxed hop constraints, Dijkstra remains at 1.000, greedy forwarding reaches 0.984 ± 0.008, and PPO, A2C, and DQN achieve high best-checkpoint success rates of approximately 0.991–0.992. REINFORCE improves under the relaxed setting but remains less stable than the stronger learned policies. The experiments show that feasibility-aware learning can approach deterministic baseline reliability while retaining learned forwarding capability under hop constraints. The ablation results further confirm that action masking is the dominant mechanism for maintaining feasible routing decisions, whereas trust mainly provides reliability-aware regularization. These observations emphasize the need to evaluate RL-based routing using deployment-level metrics, including success probability, hop-count distribution, invalid-action rate, route-risk rate, and return, rather than relying only on training reward.

KEYWORDS: Wireless sensor networks; reinforcement learning; routing protocols; hop-constrained routing; reliability-aware routing

1 Introduction

Wireless Sensor Networks (WSNs) form a key technology enabler for broad-scale monitoring and control applications such as environmental monitoring via Internet of Things (IoT) technologies, automation, and surveillance systems. In such networks, sensing nodes have limited energy, computation, and communication resources; therefore, routing plays a critical role [1]. Time-critical applications often require hop-constrained packet delivery and bounded path lengths [2]. Thus, hop constraints play a critical role in determining routing feasibility, delivery reliability, and the amount of consumed energy.

A number of previous works have considered classical routing techniques for WSNs in the literature. Low computational complexity and localized operation make greedy forwarders attractive candidates. Nevertheless, they can suffer from poor path quality or even routing failures when operating in sparse or irregular topologies [3]. Conversely, while shortest path algorithms, such as Dijkstra routing, yield globally optimal routing decisions with respect to hop count or cost functions using globally-known network data, their applicability hinges on centralized information availability and static assumptions or frequent signaling, limiting their adaptability and scaling in real-world scenarios of dynamic nature [4]. Moreover, traditional methods generally lack the ability to adapt their routing decisions to changing network states, such as residual energy.

The advancements in reinforcement learning (RL) algorithms have spurred interest in designing learning-based approaches to routing in wireless networks. Routing is viewed as a sequential decision-making process in which the agent can learn a suitable forwarding strategy by directly interacting with the environment without having a mathematical model that describes the link quality or traffic patterns [5]. In early works, it was shown that RL techniques can be utilized to perform packet routing in time-varying networks [6]. The evolution of deep learning (DL) in RL has paved the way for RL to be applied to various challenging networking problems [7].

Regarding WSN applications, many researchers have explored the use of Q-learning, deep RL, and policy gradient algorithms for routing, scheduling, and resource allocation tasks [8,9]. These studies generally achieve better adaptability and performance than traditional static routing schemes, particularly in dynamic and non-stationary settings. Nevertheless, there still exist major concerns about the stability of convergence, sparse rewards, and comparing among different learning approaches, particularly in environments where the action space is tightly constrained or dynamically masked.

Despite all these advances, there still remain some significant open research issues. Firstly, numerous current works analyze learning-based approaches to routing under relatively free circumstances without consideration of any constraints on resources and path lengths. However, resource limits and hop budgets directly influence the routing decision process because they define feasible forwarding actions, termination conditions, and deployment-level reliability requirements. Secondly, comprehensive comparisons between various reinforcement learning techniques like value-based learning, policy gradients, and actor-critic algorithms are rather rare. In addition, such experiments are usually performed using different assumptions that do not allow one to draw appropriate conclusions on applicability of the approaches discussed [10,11]. Finally, while training-related criteria such as reward are typically utilized, less attention is paid to performance measures based on delivery ratio and hop count distributions which more correspond to routing tasks.

Motivated by these gaps, this paper focuses on feasibility-aware learning for hop-restricted WSN routing, where forwarding decisions must satisfy finite-hop delivery, energy, and reliability requirements under dynamic network conditions.

The major contributions of this paper are summarized as follows:

- We formulate hop-constrained WSN routing as a finite-horizon Markov Decision Process (MDP) with an augmented state representation that includes remaining-hop budget, candidate-neighbor features, and path-memory information for loop avoidance.
- We design a feasibility-aware RL framework that separates hard routing constraints from soft trust- and energy-aware reliability terms through action masking, reward shaping, and candidate-level state encoding.
- We provide a systematic multi-seed comparison of REINFORCE, Advantage Actor–Critic (A2C), Proximal Policy Optimization (PPO), and Deep Q-Network (DQN) against greedy and Dijkstra baselines under strict and relaxed hop constraints.
- We add component ablation and reward–success correlation analyses to quantify the roles of action masking, trust filtering, trust reward, and training-reward alignment with deployment-level routing performance.

The remaining sections of the paper are structured as follows. Section 2 covers the most recent contributions concerning RL-based routing in WSNs. Section 3 provides the system model and the problem statement. Section 4 explains the adopted baseline and RL-based algorithms and integrates them into the routing scheme. Section 5 introduces the experiment setup and parameter tuning process. Section 6 discusses the numerical results and their interpretation. Section 7 highlights some limitations and further research directions. Lastly, Section 8 concludes the paper.

2 Related Work

RL has recently emerged as an attractive paradigm for adaptive routing in wireless sensor networks, which are often difficult to handle through static routing because of their dynamic topologies, energy constraints, and limited capabilities of sensor nodes [12,13]. This is because RL-based routing has been capable of enabling learning of routes through interaction with the dynamic environment.

The relative efficiency of energy use has been the most explored use of RL within WSN routing problems. The initial study highlighted the applicability of RL to packet routing within dynamically varying networks and formulated key concepts for place and route-based routing designs [6]. The use of optimized tabular Q-learning algorithms for routing designs based on residual energy and link conditions was explored within later studies. The significance of considering overall energy use on multi-hop routes was highlighted within initial designs for energy-aware routing, including [14], paving the way for emerging designs focused on route learning based on routing algorithms. As part of continuing developments within this area, research [15] designed an energy-efficient IoT-focused WSN routing algorithm using RL.

The use of deep RL has, in addition, improved routing flexibility with richer representation and more expressive policies. Applying deep Q-learning in energy harvesting WSNs, the authors in [16] presented better end-to-end throughput and energy use over heuristic methods but pointed out drawbacks in dealing with sparsity in rewards and stability when hard constraints or short time-horizons apply to routing functions. There is, thus, a need to explore other paradigms in RL, aside from value function approaches, for constrained routing optimization.

In addition to optimizing energy, the aspect of trust and security in routing using the RL method has been considered to deal with the unreliability or malperformance of nodes. In [17], the authors presented a trust-aware RL-based routing scheme in mission-critical WSNs by including trust values in the RL algorithm to enhance reliability and resistance to malicious nodes. Likewise, the work of [18] presented a federated deep RL algorithm to integrate energy efficiency and trust-aware routing, where the knowledge of the routing

algorithm is updated from the local to the global levels after a period of time. It is clear from the examples above that trust-sensitive learning leads to an increase in robustness; however, such techniques are normally applied in unconstrained routing systems without addressing the hop-budget constraints.

For overcoming issues related to dynamic topology and scalability, new research has focused on developing multi-agent systems and actor-critic architectures for reinforcement learning. For example, in [11], the authors designed a multi-agent deep reinforcement learning routing scheme that uses actor-critic algorithms to facilitate quick convergence during topology variations in tactical mobile sensor networks. Their findings reveal that their model outperforms conventional value-based reinforcement learning algorithms in terms of faster reconvergence and higher throughput. In general, policy gradient algorithms and actor-critic methods provide more stability for reinforcement learning in dynamic routing scenarios [19,20]. By directly optimizing stochastic policies and estimating advantage functions, these methods better handle sparse terminal rewards and changing action feasibility, which are common in constrained routing problems.

While existing studies demonstrate the potential of RL for energy-efficient, secure, and adaptive routing in WSNs, several gaps remain. Most prior work considers unconstrained or loosely constrained routing scenarios, with limited attention to strict hop-budget enforcement. Moreover, comparative evaluations across different RL paradigms are often conducted under heterogeneous assumptions, making it difficult to assess their relative strengths under identical constraints and evaluation criteria. Table 1 summarizes the comparison and highlights the remaining gap in strict hop-constrained RL routing.

Table 1: Comparison of representative RL-based routing approaches in wireless sensor networks.

Work	RL Type	Energy-Aware	Trust-Aware	Hop Constraint	Topology Adaptation
[6]	Tabular RL	No	No	No	Yes
[14]	Non-RL (Heuristic)	Yes	No	No	Limited
[15]	Q-learning	Yes	No	No	Yes
[16]	DQN	Yes	No	No	Limited
[17]	Deep reinforcement learning (DRL)	Yes	Yes	No	Yes
[18]	Federated DRL	Yes	Yes	No	Yes
[11]	Multi-Agent DRL	Yes	No	No	Yes
This work	**Value-based/policy-gradient/actor–critic**	**Yes**	**Lightweight**	**Yes**	**Yes**

The present work adopts a RL formulation that directly learns hop-constrained routing policies through interaction with the network environment. By providing a unified comparison of value-based, policy-gradient, and actor–critic methods under identical hop constraints, trust-aware feasibility filtering, and statistically grounded evaluation metrics, this study offers new insights into the suitability of different RL paradigms for hop-constrained routing in wireless sensor networks.

3 System Model and Problem Formulation

We consider a wireless sensor network modeled as a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{1, 2, \ldots, N\}$ denotes the set of sensor nodes and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ represents the set of communication links. An edge $(i, j) \in \mathcal{E}$ indicates that node i can directly forward a packet to node j. This graph-based abstraction is widely adopted for modeling multi-hop wireless sensor networks [12,13]. The adjacency relationship is described by a binary matrix $A \in \{0,1\}^{N\times N}$, where $A_{ij} = 1$ if $(i, j) \in \mathcal{E}$. Fig. 1 illustrates the considered WSN routing scenario, where forwarding decisions are made over candidate neighbor nodes within communication range while trust-aware routing avoids unreliable or malicious nodes.

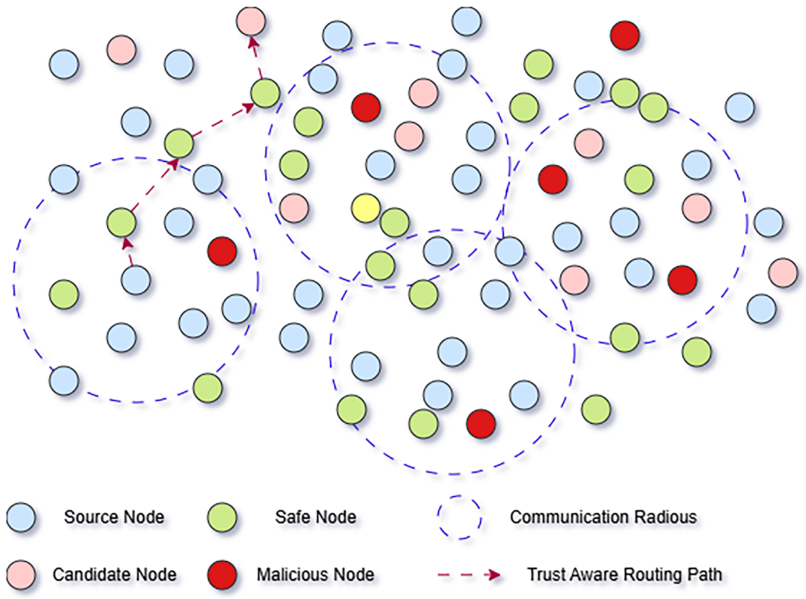

Figure 1: System model of the proposed hop-constrained and reliability-aware WSN routing framework.

Each routing episode corresponds to the delivery of a single packet from a source node $s \in \mathcal{V}$ to a destination node $d \in \mathcal{V}$. Packet forwarding is subject to a strict hop budget H, which captures latency and reliability requirements commonly imposed in time-sensitive WSN applications. A routing attempt is considered successful if the destination is reached within at most H hops; otherwise, the episode terminates unsuccessfully. Let v_t denote the node holding the packet at step t, with $v_0 = s$. At each step, the forwarding decision selects a next-hop node from the neighbor set

$$\mathcal{N}(i) = \{j \in \mathcal{V} \mid (i, j) \in \mathcal{E}\}. \tag{1}$$

The episode terminates when $v_t = d$, when $t = H$, or when no feasible forwarding action exists.

In addition to hop constraints, the routing framework incorporates a lightweight reliability indicator to reflect the historical forwarding behavior of sensor nodes. Such reliability or trust modeling has been widely adopted in WSN routing to capture node dependability without introducing explicit adversarial assumptions [17]. Each node j maintains a trust value $T_j(t) \in [0,1]$, representing its estimated reliability at time t. Trust values are updated using an exponentially weighted moving average based on observed forwarding outcomes,

$$T_j(t+1) = (1-\alpha)T_j(t) + \alpha \cdot \mathbb{I}_{\text{succ}}(j), \tag{2}$$

where $\alpha \in (0,1)$ is a smoothing factor and $\mathbb{I}_{\text{succ}}(j)$ is an indicator equal to one if node j successfully forwards a packet and zero otherwise. In the proposed trust-update model, $\mathbb{I}_{\text{succ}}(j)$ is determined from forwarding confirmation. A successful forwarding observation is recorded as $\mathbb{I}_{\text{succ}}(j) = 1$ when the forwarding node receives a link-layer acknowledgment (ACK), passive ACK, or overheard forwarding confirmation within the predefined timeout interval. If no confirmation is received within this interval, or if the forwarding attempt results in route failure, the observation is recorded as $\mathbb{I}_{\text{succ}}(j) = 0$. Trust is treated as a soft constraint during routing, such that nodes with persistently low trust values are gradually deprioritized while exploration capability is preserved.

The hop-constrained and reliability-aware routing problem is formulated as a finite-horizon MDP $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, P, r, \gamma, H \rangle$, providing a principled framework for sequential decision-making under uncertainty [5]. The state at time t, denoted $s_t \in \mathcal{S}$, encodes the current routing context and is defined as

$$s_t = \phi(v_t, d, h_t, \mathcal{G}, \mathcal{V}_t^{\text{vis}}, \mathcal{T}_t), \tag{3}$$

where $\phi(\cdot)$ is a feature-mapping function, v_t is the current forwarding node, d is the destination node, h_t denotes the remaining-hop budget, $\mathcal{G}$ is the network graph, $\mathcal{V}_t^{\text{vis}}$ denotes the set of nodes already visited in the current route, and $\mathcal{T}_t$ denotes the trust values associated with candidate neighbor nodes. The feature-mapping function transforms the variable-size local routing state into a fixed-length input vector suitable for neural RL agents. The raw routing state contains graph-dependent information, including a variable number of candidate neighbors and path-memory information. The mapping therefore encodes current-node status, destination information, remaining-hop budget, visited-path information, candidate-neighbor features, and the action mask. This representation makes loop-avoidance information explicit rather than treating it as hidden history, while allowing all learning algorithms to use a common observation dimension.

The action space is defined over a fixed maximum cardinality K, where K is chosen to be greater than or equal to the maximum node degree. Each action corresponds to selecting one candidate neighbor. To enforce feasibility, an action mask $m_t \in \{0,1\}^K$ is applied, where $m_t(k) = 1$ indicates a valid forwarding action. The effective action set is therefore

$$\mathcal{A}(s_t) = \{a \in \{1, \ldots, K\} \mid m_t(a) = 1\}. \tag{4}$$

State transitions are deterministic given the selected action and the network topology, mapping (s_t, a_t) to s_{t+1} by updating the current node, remaining hop budget, and trust estimates. We distinguish hard routing feasibility constraints from soft reliability-aware terms. Connectivity, positive residual energy, valid candidate selection, no-loop routing, and hop-budget feasibility are treated as hard constraints because they determine whether an action can physically produce a feasible route transition. Trust, in contrast, is treated as a reliability-aware soft component that is used for reward regularization or optional candidate screening rather than as a physical feasibility condition. Feasibility is incorporated in the RL framework using action masking, ensuring that decisions are restricted to forwarding actions satisfying the hard constraints. This separation keeps the learning process well posed under strict hop restrictions while allowing the trust component to be examined independently through ablation analysis. This design is consistent with recent feasibility-aware RL frameworks for communication-constrained networks [21].

The reward function is designed to promote successful delivery within the hop constraint while discouraging inefficient or unreliable routing decisions,

$$r_t = \begin{cases} R_{\text{succ}}, & \text{if } v_{t+1} = d, \\ -R_{\text{fail}}, & \text{if } t+1 = H \text{ and } v_{t+1} \neq d, \\ -R_{\text{dead}}, & \text{if } \mathcal{A}(s_{t+1}) = \varnothing, \\ -\eta - \lambda(1 - T_{v_{t+1}}), & \text{otherwise}, \end{cases} \tag{5}$$

where $R_{\text{succ}} > 0$ represents the terminal success reward, $R_{\text{fail}} > 0$ and $R_{\text{dead}} > 0$ penalize hop-budget violations and dead-end states, $\eta > 0$ represents the energy penalty per hop, $\lambda > 0$ is the weight assigned to trust in route selection, and $T_{v_{t+1}}$ is the trust value of the selected next-hop node. The trust and energy functions used herein are deliberately designed to be simplistic in nature. The reason behind this consideration lies in the resource-constrained nature of WSN nodes, where sophisticated trust models and energy consumption estimates are impractical. The trust indicator is not intended to capture sophisticated attack strategies,

but rather to provide a minimal reliability signal based on historical forwarding behavior, enabling the interaction between trust-awareness and hop-constrained learning dynamics to be examined in a controlled setting. Similarly, energy consumption is modeled implicitly through hop count and per-hop penalties, which is a widely adopted abstraction in WSN routing analysis.

The return for an episode is defined as

$$G_0 = \sum_{t=0}^{T-1} \gamma^t r_t, \tag{6}$$

where $T \leq H$ is the termination time. The objective is to learn a policy $\pi_\theta(a|s)$ that maximizes the expected return,

$$J(\theta) = \mathbb{E}_{\pi_\theta}[G_0]. \tag{7}$$

Routing performance is evaluated using delivery success rate, average hop count, cumulative return, and hop-count cumulative distribution functions (CDF). For multi-seed experiments, 95% confidence intervals are reported as

$$\mu \pm 1.96 \frac{\sigma}{\sqrt{S}}, \tag{8}$$

where μ and σ denote the mean and standard deviation across S independent runs, following best practices in RL-based routing evaluation [10,11].

4 Routing Schemes and Learning Algorithms

This section describes all routing mechanisms evaluated in this study, including classical baselines and RL-based approaches. All schemes are evaluated under identical topologies, hop constraints, and candidate-neighbor definitions. Hard feasibility is determined by connectivity, residual energy, loop avoidance, and remaining hop budget, while trust is treated as a reliability-aware screening or shaping term and examined separately through ablation. The routing decision at each step selects a feasible next-hop neighbor from the current node.

4.1 Greedy Forwarding

Greedy forwarding is a distributed heuristic in which each node selects the neighbor that minimizes a local distance metric to the destination. At routing step t, the next hop is chosen as

$$a_t^{\text{greedy}} = \arg \min_{j \in \mathcal{N}(v_t)} D(j, d), \tag{9}$$

where $D(j, d)$ denotes the estimated minimum hop distance from node j to the destination d, computed using a breadth-first search (BFS) tree rooted at d.

For consistency with the proposed framework, greedy forwarding applies the same feasibility and trust-screening rules used in the main routing configuration. Among the admissible candidates, it selects the neighbor with the smallest estimated hop distance to the destination; if no neighbor satisfies the trust-screening rule, the feasible neighbor with the highest trust value is selected.

Greedy forwarding imposes virtually zero computational cost, but it is myopic by nature and could get stuck in local minimum cases. Algorithm 1 summarizes the greedy forwarding procedure.

Algorithm 1: Greedy forwarding

1: **Input:** current node v_t, destination d, hop budget H
2: Compute feasible set $\mathcal{N}_{\text{greedy}}(v_t)$ with $T_j \geq T_{\min}$
3: **if** $\mathcal{N}_{\text{greedy}}(v_t) = \varnothing$ **then**
4: Select neighbor with maximum trust
5: **else**
6: Select $j = \arg\min D(j, d)$
7: **end if**
8: Forward packet to j

4.2 Dijkstra Shortest-Path Routing

The Dijkstra baseline constructs a shortest path using full information about the network topology and unit edge costs, and therefore serves as a centralized reference baseline for the distributed learning policies. The effective graph $\mathcal{G}_{\text{eff}}$ is obtained by applying the same hard feasibility and configured trust-screening rules used in the main comparison. Trust screening is therefore used only as a reliability-aware candidate-filtering rule, not as a physical connectivity constraint. Let $\boldsymbol{\mathcal{P}}^{\star} = (v_0 = s, \ldots, v_L = d)$ be the resulting shortest path. If $L \leq H$, then packets are routed hop-by-hop along this path; otherwise, routing fails because of hop infeasibility. Algorithm 2 summarizes the Dijkstra-based routing procedure.

Algorithm 2: Dijkstra routing

1: **Input:** graph $\mathcal{G}$, source s, destination d, hop budget H
2: Construct effective graph $\mathcal{G}_{\text{eff}}$ using hard feasibility and configured trust-screening rules
3: Compute shortest path $\boldsymbol{\mathcal{P}}^{\star} = (v_0 = s, \ldots, v_L = d)$ on $\mathcal{G}_{\text{eff}}$ using Dijkstra
4: **if** $L \leq H$ **then**
5: Forward packet along $\boldsymbol{\mathcal{P}}^{\star}$
6: **else**
7: Declare routing failure
8: **end if**

Although Dijkstra guarantees hop-optimal routing on the effective graph, the method requires global topology knowledge and repeated path computation, which limits scalability in dynamic WSNs.

4.3 Policy Gradient Routing (REINFORCE)

REINFORCE learns a stochastic routing policy $\pi_\theta(a|s)$ by maximizing the expected return using Monte Carlo trajectories. The policy update is given by

$$\nabla_\theta J(\theta) = \mathbb{E}\left[\sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t|s_t)\,(G_t - b(s_t))\right], \tag{10}$$

where $b(s_t)$ is a variance-reducing baseline.

Action feasibility is enforced through masked softmax over candidate neighbors. Trust information influences learning through reward shaping and state features.

REINFORCE is simple to implement but exhibits high variance and slow convergence under hop-constrained termination. Algorithm 3 summarizes the REINFORCE routing procedure.

Algorithm 3: REINFORCE routing

1: **for** each episode **do**
2: Sample routing trajectory using masked policy π_θ
3: Compute returns G_t
4: Update θ using policy gradient
5: **end for**

4.4 Advantage Actor–Critic (A2C)

A2C introduces a critic network to estimate the value function $V_\phi(s)$, enabling lower-variance updates. The advantage is computed as

$$A_t = r_t + \gamma V_\phi(s_{t+1}) - V_\phi(s_t). \tag{11}$$

The actor and critic are updated jointly using temporal-difference (TD) learning. Algorithm 4 summarizes the A2C routing procedure.

Algorithm 4: A2C routing

1: **for** each episode **do**
2: Collect transitions (s_t, a_t, r_t, s_{t+1})
3: Compute advantages A_t
4: Update actor using A_t
5: Update critic via TD error
6: **end for**

A2C provides improved learning stability and faster convergence compared with REINFORCE.

4.5 Proximal Policy Optimization (PPO)

PPO constrains policy updates using a clipped surrogate objective

$$\mathcal{L}_{\text{PPO}} = \mathbb{E}\left[\min\left(\rho_t A_t, \text{clip}(\rho_t, 1-\varepsilon, 1+\varepsilon)A_t\right)\right]. \tag{12}$$

The conservative update mechanism makes PPO particularly effective under hop constraints. Algorithm 5 summarizes the PPO routing procedure.

Algorithm 5: PPO routing

1: **for** each iteration **do**
2: Collect trajectories with current policy
3: Estimate advantages using generalized advantage estimation (GAE)
4: Perform clipped policy updates
5: **end for**

4.6 Deep Q-Network (DQN)

DQN learns an action-value function $Q_\psi(s, a)$ using TD updates with experience replay. Action masking is applied during both action selection and target computation.

DQN struggles under hop-constrained routing due to sparse rewards and limited value propagation. Algorithm 6 summarizes the DQN routing procedure.

Algorithm 6: DQN routing

1: **for** each step **do**
2: Select action via masked ε-greedy policy
3: Store transition in replay buffer
4: Update Q_ψ using minibatch TD loss
5: **end for**

5 Simulation Setup and Parameter Configuration

This section outlines the simulation setting and parameter values for testing the proposed hop-constrained and reliability-aware routing strategy. All parameter values are reported based on the implemented model to allow for reproducible and comparable evaluation across different routing strategies.

The WSN is composed of $N = 200$ sensor nodes that are evenly scattered within a two-dimensional 100×100 space. Network connectivity is achieved using a predefined communication range, leading to a geometric graph model frequently adopted in WSN research [12,13]. Every routing attempt involves the delivery of one packet from an arbitrarily chosen source to a randomly chosen destination under a strict hop-budget constraint. Two hop-budget scenarios are defined to represent tight and loose latency constraints.

At each hop of the routing process, routing actions are taken among a predetermined number of neighboring nodes to keep the action space dimension constant for all learning agents. The reliability feature is integrated using a lightweight trust system that provides a reliability-aware screening and reward-shaping signal, without assuming an explicit malicious attack model.

Table 2 presents a detailed summary of all simulation, routing, and learning-related parameters.

Table 2: Simulation parameters and configuration settings.

Parameter	Value/Description	Parameter	Value/Description
Network area	100×100 units	Number of nodes (N)	200
Node deployment	Uniform random	Communication radius	15 units
Graph model	Geometric graph	Routing episode	Single packet per episode
Hop budget (H)	5 (strict), 15 (relaxed)	Minimum hop diagnostic	BFS shortest hop count
Candidate neighbors (K)	20	Hard feasibility	Connectivity, energy, hop budget, no loop
Action feasibility	Binary action mask	Trust handling	Soft reward/optional screening, $T_{\min} = 0.5$
Observation vector	Energy, trust, hops, visited state, mask	Forwarding confirmation	ACK-style success proxy
Failure observations	Timeout, invalid forwarding, depletion	Path-memory encoding	Visited fraction and candidate visited flags
Initial node energy	$P = 1.0$	Energy depletion	Episode termination
Transmit energy cost	0.0015	Receive energy cost	0.0006
Idle energy cost	0.0002	Trust update	Energy- and reliability-weighted
Initial trust value	0.7	Trust decay factor	0.9995 (inactive nodes)
Success reward	+2.0	Failure penalty	−2.0
Step penalty	−0.002	Trust shaping	Reliability-aware reward regularization

(Continued)

Table 2 (continued)

Parameter	Value/Description	Parameter	Value/Description
RL algorithms	REINFORCE, A2C, PPO, DQN	Hidden layers	2 layers, 128 units each
Discount factor (γ)	0.99	Entropy coefficient	0.01
Learning rate (PG/AC)	3×10^{-4}	Learning rate (DQN)	10^{-3}
PPO GAE parameter (λ)	0.95	PPO clip parameter (ε)	0.2
Replay buffer size (DQN)	10^5	Batch size (DQN)	256
Target update interval (DQN)	250 steps	Random seeds	5 independent runs
Performance metrics	Success, hops, return, hop CDF	Confidence interval	95% across seeds

In the implemented simulations, the observation vector includes the current-node energy and trust states, destination information, the remaining-hop ratio, the visited-node fraction, candidate-neighbor features, candidate-level visited indicators, and action-mask entries. This augmented observation makes the path-memory information required for loop avoidance explicit during RL training and evaluation. Forwarding confirmation is implemented through an ACK-style success proxy based on receiver availability and link-quality satisfaction, while timeout, invalid forwarding, and energy-depletion events are treated as unsuccessful forwarding observations.

All routing mechanisms are run under equal feasibility conditions, trust thresholds, and hop constraints in order to ensure that any difference in the performance observed is purely due to the routing strategy.

6 Results and Discussion

In this section, we evaluate learning-based routing schemes against conventional baselines for hop-constrained packet forwarding. The experiments are conducted using multiple trials across five independent seeds. Results are reported for two cases: a strict hop budget of $H = 5$ and a relaxed hop budget of $H = 15$. Delivery success probability, hop efficiency, cumulative return, and learning dynamics are analyzed, as these metrics jointly capture the deployment level reliability and training stability.

6.1 Delivery Success Rate

Fig. 2 compares delivery success probability and hop-count distributions under the two hop-budget regimes. For the strict case $H = 5$, Dijkstra reaches a success rate of 1.000, while greedy forwarding reaches 0.950 ± 0.014. Among the learning algorithms, PPO, A2C, and DQN each reach 0.945 ± 0.014 at their best checkpoints, whereas REINFORCE remains lower at 0.753 ± 0.027. The last-checkpoint results show that A2C and DQN preserve their best-checkpoint reliability more consistently than PPO, whose final success rate decreases to 0.834 ± 0.023.

When the hop budget is relaxed to $H = 15$, the success rates increase for most methods. Dijkstra remains at 1.000, greedy forwarding reaches 0.984 ± 0.008, and PPO, A2C, and DQN achieve 0.992 ± 0.006, 0.991 ± 0.006, and 0.992 ± 0.006 at their best checkpoints, respectively. REINFORCE also improves to 0.955 ± 0.013 at its best checkpoint but decreases to 0.911 ± 0.018 at the last checkpoint, indicating higher training variability. Overall, the results show that feasibility-aware action selection enables the stronger learned policies to approach the Dijkstra reference baseline while using learned local forwarding decisions.

6.2 Hop Efficiency and Hop-Count Distributions

The hop-count CDFs in Fig. 2, together with the average hop counts in Fig. 3, characterize routing efficiency. Under $H = 5$, hop counts are tightly concentrated because the strict hop budget limits feasible

routing paths. Dijkstra and greedy forwarding achieve mean hop counts of 4.33 ± 0.04 and 4.33 ± 0.04, respectively. PPO, A2C, and DQN remain in the same range at their best checkpoints, with average hop counts around 4.34 ± 0.04. REINFORCE has lower delivery reliability, so its hop-count behavior should be interpreted together with its reduced success rate.

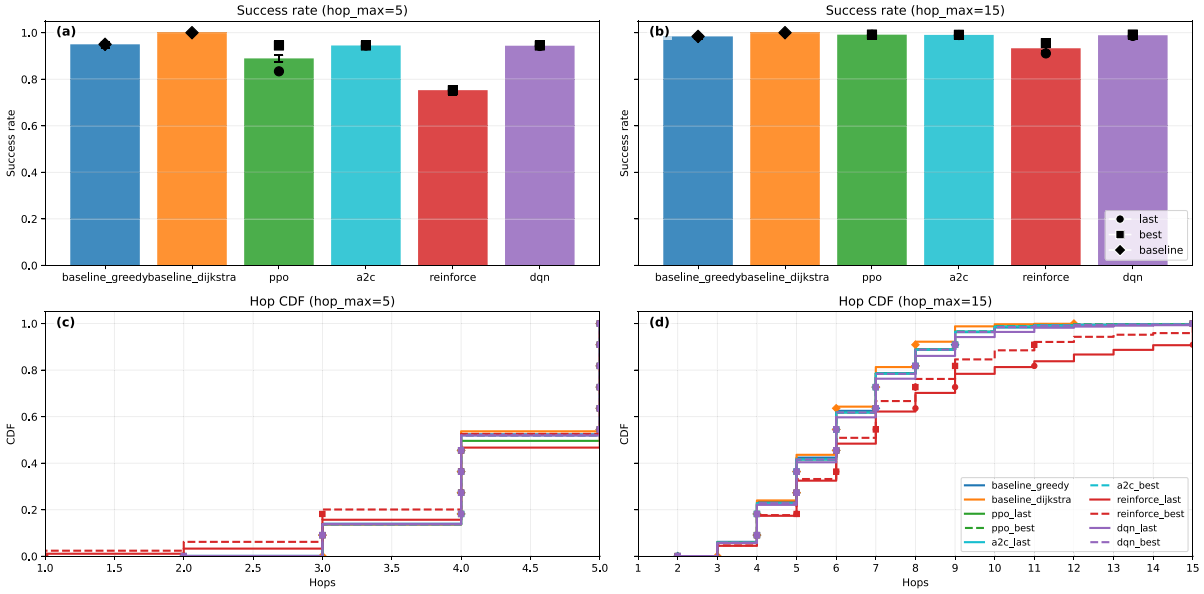

Figure 2: Delivery success rate and hop-count CDFs under strict and relaxed hop constraints: (**a**) success rate for $H = 5$; (**b**) success rate for $H = 15$; (**c**) hop-count CDF for $H = 5$; and (**d**) hop-count CDF for $H = 15$. Bars summarize method families, circles and squares denote last and best RL checkpoints, and diamonds denote fixed baselines.

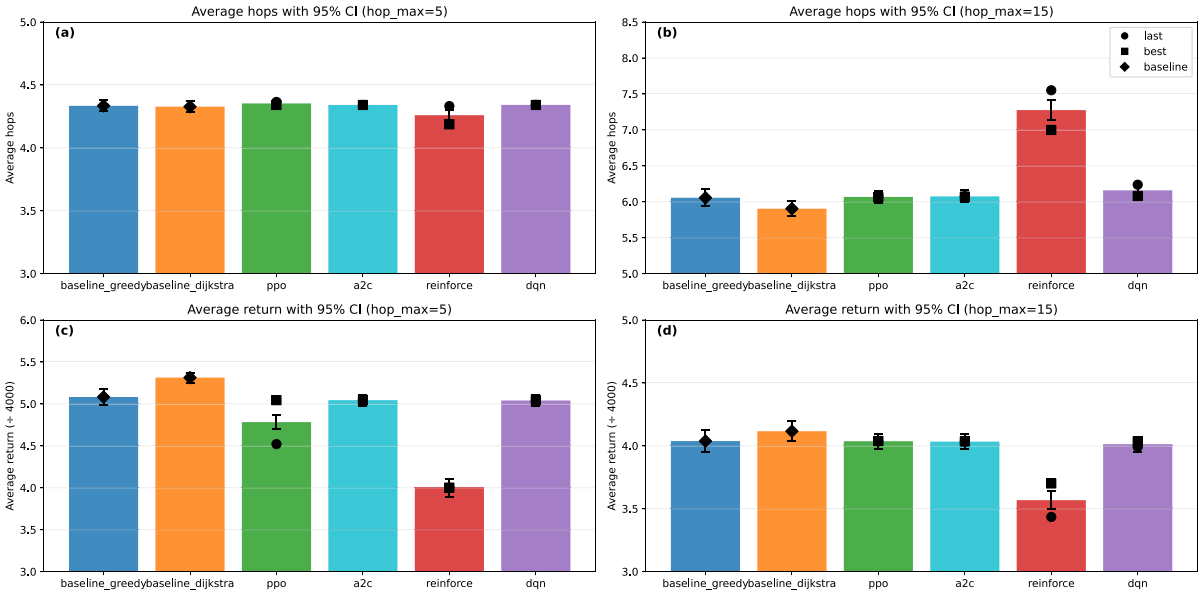

Figure 3: Average hop count and scaled cumulative return under strict and relaxed hop constraints: (**a**) average hop count for $H = 5$; (**b**) average hop count for $H = 15$; (**c**) scaled cumulative return for $H = 5$; and (**d**) scaled cumulative return for $H = 15$. Error bars indicate 95% confidence intervals, and return values are divided by 4000 for compact visualization.

Under the relaxed constraint $H = 15$, routing-efficiency differences become more visible. Dijkstra achieves an average hop count of 5.90 ± 0.11, while greedy forwarding reaches 6.06 ± 0.12. PPO, A2C, and DQN remain close to these baselines, with best-checkpoint average hop counts of 6.06 ± 0.12, 6.07 ± 0.12, and 6.08 ± 0.12, respectively. In contrast, REINFORCE records a higher best-checkpoint average hop count of approximately 7.00 ± 0.17. These results indicate that the stronger learned policies maintain high delivery reliability without substantially increasing route length.

6.3 Cumulative Return

Fig. 3 also reports the scaled cumulative return. For $H = 5$, Dijkstra obtains the highest scaled return of 5.31 ± 0.06, followed by greedy forwarding at 5.08 ± 0.09. PPO, A2C, and DQN remain close to the greedy baseline at their best checkpoints, with scaled returns of approximately 5.04 ± 0.09. REINFORCE is lower at approximately 4.00 ± 0.15, reflecting its lower delivery success rate.

For $H = 15$, Dijkstra and greedy forwarding achieve scaled returns of 4.12 ± 0.08 and 4.04 ± 0.09, respectively. PPO, A2C, and DQN produce comparable best-checkpoint scaled returns of approximately 4.04, while REINFORCE remains lower at approximately 3.70 ± 0.09. These trends are consistent with the success-rate and hop-efficiency results, showing that the stronger learned policies provide a favorable balance between delivery reliability and routing efficiency.

6.4 Learning Dynamics and Convergence

Fig. 4 presents the learning dynamics of the four RL algorithms using four panels: normalized training reward for $H_{\max} = 5$ and $H_{\max} = 15$, and evaluation success rate for the same two hop constraints. In each panel, the faint curves represent the unsmoothed across-seed averaged trajectories, while the bold curves represent exponentially smoothed trends. Specifically, the bold trends are obtained using exponential moving average smoothing, with span 15 for the reward curves and span 8 for the evaluation-success curves. The figure shows that increases in training reward do not always translate monotonically into higher deployment success. A2C reaches stable high evaluation success early, while PPO achieves strong performance under the relaxed hop constraint but shows last-checkpoint degradation under the stricter $H_{\max} = 5$ setting. DQN improves more gradually and eventually reaches high evaluation success, whereas REINFORCE remains less stable, particularly under the relaxed hop setting. This reward–success relationship is quantified later using Pearson and Spearman correlation coefficients, rather than being inferred only from inspection of the learning curves.

Table 3 complements Fig. 4 by reporting deployment-level performance across multiple seeds. The results show that routing reliability must be evaluated using task-specific metrics, not training reward alone. Therefore, the revised analysis reports success rate, hop count, route-risk rate, invalid-action rate, return, and reward–success correlation together to provide a quantitative basis for the claim that training reward alone is insufficient for assessing routing performance.

6.5 Component Ablation Study

To quantify the effect of the proposed components, we conducted an ablation study with five variants: the full model, removal of the trust-reward term ($\lambda = 0$), removal of trust-based filtering, removal of both trust reward and trust filtering, and removal of feasibility-aware action masking. Table 4 summarizes the results for A2C and PPO, which are the two actor–critic methods emphasized in the reviewer-requested ablation.

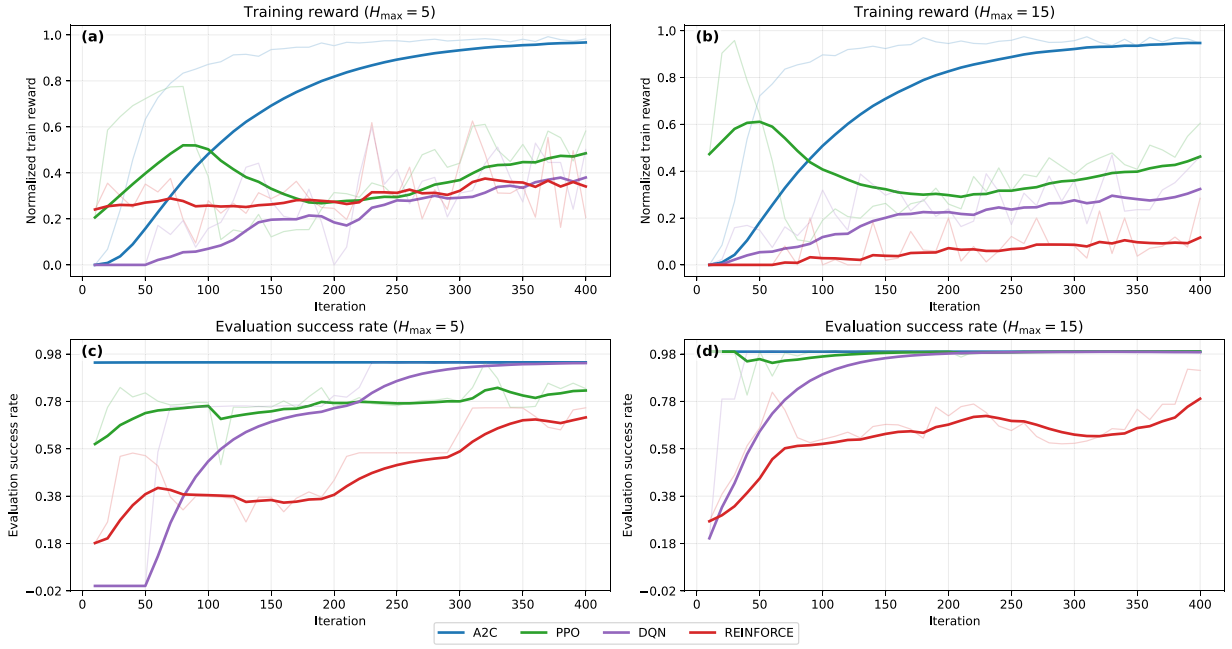

Figure 4: Learning dynamics of the RL algorithms under strict and relaxed hop constraints: (**a**) normalized training reward for $H_{max} = 5$; (**b**) normalized training reward for $H_{max} = 15$; (**c**) evaluation success rate for $H_{max} = 5$; and (**d**) evaluation success rate for $H_{max} = 15$.

Table 3: Across-seed evaluation summary (mean ± 95% confidence interval) under strict ($H = 5$) and relaxed ($H = 15$) hop constraints.

Scheme	Strict Hop Constraint ($H = 5$)			Relaxed Hop Constraint ($H = 15$)		
	Success	Hops	Return/4000	Success	Hops	Return/4000
Greedy (baseline)	0.950 ± 0.014	4.334 ± 0.044	5.082 ± 0.092	0.984 ± 0.008	6.055 ± 0.119	4.038 ± 0.086
Dijkstra (baseline)	1.000 ± 0.000	4.327 ± 0.043	5.314 ± 0.058	1.000 ± 0.000	5.903 ± 0.105	4.116 ± 0.080
PPO (best)	0.945 ± 0.014	4.341 ± 0.044	5.045 ± 0.094	0.992 ± 0.006	6.062 ± 0.116	4.038 ± 0.083
PPO (final)	0.834 ± 0.023	4.365 ± 0.044	4.521 ± 0.136	0.991 ± 0.006	6.072 ± 0.118	4.035 ± 0.084
A2C (best)	0.945 ± 0.014	4.341 ± 0.044	5.045 ± 0.094	0.991 ± 0.006	6.072 ± 0.118	4.035 ± 0.084
A2C (final)	0.945 ± 0.014	4.341 ± 0.044	5.045 ± 0.094	0.990 ± 0.006	6.077 ± 0.119	4.032 ± 0.084
REINFORCE (best)	0.753 ± 0.027	4.187 ± 0.060	4.001 ± 0.151	0.955 ± 0.013	6.999 ± 0.175	3.701 ± 0.094
REINFORCE (final)	0.753 ± 0.027	4.332 ± 0.053	4.001 ± 0.151	0.911 ± 0.018	7.551 ± 0.213	3.435 ± 0.107
DQN (best)	0.945 ± 0.014	4.341 ± 0.044	5.046 ± 0.094	0.992 ± 0.006	6.080 ± 0.119	4.036 ± 0.084
DQN (final)	0.943 ± 0.014	4.342 ± 0.044	5.036 ± 0.095	0.986 ± 0.007	6.237 ± 0.131	3.991 ± 0.086

Table 4: Component ablation results for A2C and PPO under strict and relaxed hop constraints. Values report success rate and invalid-action rate averaged across five training seeds.

Variant	Policy	H = 5		H = 15	
		Success	Invalid-Action Rate	Success	Invalid-Action Rate
Full model	A2C (best)	0.945	0.000	0.993	0.000
$\lambda = 0$	A2C (best)	0.945	0.000	0.995	0.000
No trust filter	A2C (best)	0.945	0.000	0.994	0.000

(Continued)

Table 4 (continued)

Variant	Policy	H = 5		H = 15	
		Success	Invalid-Action Rate	Success	Invalid-Action Rate
No trust	A2C (best)	0.945	0.000	0.994	0.000
No action mask	A2C (best)	0.945	0.009	0.982	0.018
Full model	PPO (best)	0.784	0.000	0.997	0.000
$\lambda = 0$	PPO (best)	0.791	0.000	0.995	0.000
No trust filter	PPO (best)	0.791	0.000	0.996	0.000
No trust	PPO (best)	0.904	0.000	0.997	0.000
No action mask	PPO (best)	0.082	0.326	0.760	0.130

The zero invalid-action rates for the full model and trust-related ablations are expected because the feasibility-aware action mask remains active in these variants and prevents infeasible next-hop selections. Therefore, nonzero invalid-action rates appear only when the action mask is removed.

The ablation results show that feasibility-aware action masking is the dominant contributor to reliable routing. Removing the action mask exposes the learning policies to infeasible next-hop selections and substantially degrades PPO performance, especially under the stricter hop constraint. For example, PPO success decreases from 0.784 to 0.082 at $H = 5$, while the invalid-action rate increases to 0.326. Under $H = 15$, PPO success decreases from 0.997 to 0.760, with an invalid-action rate of 0.130. In contrast, removing only the trust-reward term or trust-based filtering does not produce consistent degradation in the present non-adversarial simulation setting. These results indicate that trust mainly provides reliability-aware regularization and monitoring, whereas feasibility-aware action masking is the primary mechanism preventing invalid routing decisions and stabilizing policy learning.

To address the relation between training reward and deployment performance, we also computed Pearson and Spearman correlations between training reward and evaluation success rate across evaluation checkpoints. Table 5 reports the across-seed averages. Some runs reach nearly constant success rates, making the correlation mathematically undefined for those seeds; these cases are excluded from the finite-correlation average and reported in the generated correlation files.

The correlation analysis confirms that training reward and deployment success are not always strongly aligned. This supports the revised evaluation strategy, where routing-level metrics such as success rate, hop count, invalid-action rate, route-risk rate, and return are reported in addition to reward curves.

6.6 Computational Complexity and Learning Overhead

The routing algorithms considered in this research exhibit significant variation with respect to the nature of their computation processes, specifically with regard to offline computation, learning phase overheads, and deployment-related decision making. Traditional approaches depend on deterministic computation processes and do not need any learning. The greedy algorithm makes local decisions using neighborhood information, whereas Dijkstra routing performs centralized shortest-path computation that can be carried out offline for static networks.

In contrast, RL-based routing requires further computation time spent during training due to interactions with the environment. The training cost is directly dependent on the number of interactions required within an episode, which is limited by the hop constraint, and the total number of learnable parameters

within the trained models. Value-based, policy-gradient, and actor–critic approaches are characterized by differences in how they propagate their learning processes and update gradients during training. Once trained, however, all three learning approaches operate identically during deployment, where the final output route is found by evaluating the network with a single forward pass and taking feasibility into account using action masking. A summary of the asymptotic computational complexity of all investigated techniques is shown in Table 6. It should be noted that there is a clear separation between the training phase and deployment phase, which will enable us to distinguish computationally between different types of learning techniques.

Table 5: Reward–success correlation analysis across evaluation checkpoints. Values report across-seed mean Pearson and Spearman correlations for finite-correlation runs.

Hop Limit	Policy	Pearson r	Spearman ρ	Interpretation
$H = 5$	A2C	0.500	0.271	Moderate but partly saturated
$H = 5$	DQN	0.439	0.543	Moderate association
$H = 5$	PPO	0.250	0.163	Weak association
$H = 5$	REINFORCE	0.160	0.166	Weak association
$H = 15$	A2C	−0.094	−0.054	Saturated/weak association
$H = 15$	DQN	0.159	0.083	Weak association
$H = 15$	PPO	0.168	0.165	Weak association
$H = 15$	REINFORCE	−0.008	0.222	Weak and unstable association

Table 6: Computational complexity comparison of routing schemes.

Method	Training-Time Complexity	Deployment-Time Complexity
Greedy	Not applicable	$\mathcal{O}(\lvert\mathcal{N}_i\rvert)$
Dijkstra	$\mathcal{O}(\lvert\mathcal{E}\rvert + \lvert\mathcal{V}\rvert \log \lvert\mathcal{V}\rvert)$ (offline)	Path lookup
REINFORCE	$\mathcal{O}(T\lvert\theta\rvert)$	$\mathcal{O}(\lvert\theta\rvert)$
DQN	$\mathcal{O}(T\lvert\theta\rvert)$ + Q-update overhead	$\mathcal{O}(\lvert\theta\rvert)$
A2C	$\mathcal{O}(T\lvert\theta\rvert)$ + advantage estimation	$\mathcal{O}(\lvert\theta\rvert)$
PPO	$\mathcal{O}(KT\lvert\theta\rvert)$	$\mathcal{O}(\lvert\theta\rvert)$

6.7 Key Observations

Across both hop budgets, the stronger learned policies achieve near-baseline delivery reliability while maintaining comparable hop efficiency and cumulative return. A2C provides the most stable early success behavior, PPO achieves strong best-checkpoint performance but shows some last-checkpoint degradation under the stricter $H = 5$ setting, and DQN improves more gradually but reaches competitive final performance in the revised multi-seed evaluation. REINFORCE remains less stable, especially under the relaxed hop setting. These observations indicate that feasibility-aware action masking and deployment-level evaluation are central to reliable hop-constrained routing, and that training reward alone is not sufficient for selecting the best routing policy.

The routing problem formulation employed in this work is single agent and per packet, whereby each packet-forwarding operation is considered separately. Such an approach allows analyzing the effects of learning algorithms that operate under strict hop constraints without considering other elements of WSNs,

and facilitates the comparison of conventional routing schemes with various RL approaches in a well-controlled environment. Nonetheless, during practical application of wireless networks, several packets and flows could be present at any one time, causing problems like network congestion and contentions, which are important aspects of routing operations in WSNs. Although no attempts have been made in this work to model interactions among multi-flows, the developed approach is naturally extensible for addressing this issue in the future. In addition, the effects of congestion, through features like channel occupancy and flow priorities, can also be addressed using RL methods.

7 Limitations and Future Work

Although this study provides a comprehensive comparison of RL-based routing algorithms under hop constraints, several limitations remain for future investigation. First, the existing assessment is conducted on static network topology scenarios with random sensor node placement. Though such a controlled environment is required for analysis of the effect of hop constraints and learning processes, in real-world scenarios involving WSNs, there might be mobility in sensor nodes and time-varying quality of links because of environmental changes. Extension of the same framework to include these aspects would help in even better assessment of routing resilience. Such extensions can be naturally incorporated within the proposed MDP formulation by augmenting the state space with time-varying connectivity and link-quality indicators, without altering the underlying learning framework. Secondly, the reward function focuses on the successful delivery and the hops, taking into consideration the energy and trust dynamics implicitly. Future work could formulate the routing problem as a constrained or multi-objective RL task, where network lifetime, fairness among nodes, and load balancing are optimized jointly alongside delivery performance. Such formulations would better reflect long-term operational objectives in large-scale WSNs.

Thirdly, the action space has been designed using a fixed-size candidate neighbor set and a predetermined feature representation for the state. However, an alternative approach for better scalability could involve adaptive candidate neighbor selection and state representations using graph structures. The use of graph neural networks (GNNs) or attention-based encoder mechanisms could allow the agents to leverage any structural features of the network topology within dense or heterogeneous network environments. Lastly, in this work, single-agent learning problems have been considered, where a decision made by an agent affects the routing of a single flow. One area that would be important for future research concerns the design of cooperative learning frameworks involving multiple flows being simultaneously routed by the agent. Such test cases including hardware-in-the-loop experimentation and small-scale testbed implementations are key for verifying the feasibility of the proposed model. The feasibility-aware action masking mechanism used in this study, along with its finite horizon design, makes the proposed framework feasible for experimental implementations as well.

8 Conclusion

This paper investigated feasibility-aware RL for hop-constrained routing in WSNs and compared four representative RL algorithms with greedy and Dijkstra baselines under strict and relaxed hop limits. The revised multi-seed results show that the strongest learned policies approach the reliability of deterministic baselines while enabling learned forwarding decisions. For $H = 5$, Dijkstra achieves a success rate of 1.000, greedy forwarding reaches 0.950 ± 0.014, and PPO, A2C, and DQN reach approximately 0.945 ± 0.014 at their best checkpoints. For $H = 15$, PPO, A2C, and DQN achieve approximately 0.991–0.992 success at their best checkpoints, close to the Dijkstra reference baseline.

The ablation study provides a more precise interpretation of the proposed framework. Removing feasibility-aware action masking substantially degrades PPO performance, reducing success from 0.784 to

0.082 at $H = 5$ and from 0.997 to 0.760 at $H = 15$, while introducing invalid-action rates of 0.326 and 0.130, respectively. In contrast, removing only the trust reward or trust filtering produces limited degradation in the non-adversarial setting. These results indicate that action masking is the dominant mechanism for maintaining routing feasibility, whereas trust mainly provides reliability-aware regularization and monitoring. The reward–success correlation analysis further shows that training reward alone is not always a sufficient proxy for deployment reliability. Therefore, future RL-based routing studies should report deployment-level metrics, including success probability, hop-count distribution, route-risk rate, invalid-action rate, and energy-aware return, in addition to training reward.

Acknowledgement: Not applicable.

Funding Statement: This work was supported by the Deanship of Scientific Research, Vice Presidency for the Graduate Studies and Scientific Research, King Faisal University, Saudi Arabia [Grant No. KFU263841].

Author Contributions: The authors confirm their contributions to the paper as follows: Conceptualization, Adeel Iqbal and Muhammad Faisal Siddiqui; methodology, Adeel Iqbal and Muhammad Faisal Siddiqui; software, Adeel Iqbal and Muhammad Faisal Siddiqui; validation, Adeel Iqbal and Muhammad Faisal Siddiqui; formal analysis, Adeel Iqbal and Muhammad Faisal Siddiqui; investigation, Adeel Iqbal and Muhammad Faisal Siddiqui; resources, Adeel Iqbal and Muhammad Faisal Siddiqui; data curation, Adeel Iqbal and Muhammad Faisal Siddiqui; writing—original draft preparation, Adeel Iqbal and Muhammad Faisal Siddiqui; writing—review and editing, Adeel Iqbal and Muhammad Faisal Siddiqui; visualization, Adeel Iqbal and Muhammad Faisal Siddiqui; supervision, Muhammad Faisal Siddiqui; project administration, Adeel Iqbal; funding acquisition, Adeel Iqbal and Muhammad Faisal Siddiqui;. All authors reviewed approved the final version of the manuscript.

Availability of Data and Materials: The data that support the findings of this study are available from corresponding authors upon reasonable request.

Ethics Approval: Not applicable.

Conflicts of Interest: Given his role as Editorial Board Member of this journal, Adeel Iqbal had no involvement in the peer review of this article and had no access to information regarding its peer review. Full responsibility for the editorial process for this article was delegated to another journal editor. The authors declare no other conflicts of interest.

References

1. Javaid N, Qureshi TN, Khan AH, Iqbal A, Akhtar E, Ishfaq M. EDDEEC: enhanced developed distributed energy-efficient clustering for heterogeneous wireless sensor networks. Procedia Comput Sci. 2013;19:914–9.
2. Krishna KPR, Thirumuru R. Energy efficient and multi-hop routing for constrained wireless sensor networks. Sustain Comput Inform Syst. 2023;38(11):100866. doi:10.1016/j.suscom.2023.100866.
3. Benmahdi MB, Lehsaini M. Greedy forwarding routing schemes using an improved K-means approach for wireless sensor networks. Wirel Pers Commun. 2021;119(2):1619–42. doi:10.1007/s11277-021-08298-2.
4. Manev N, Temelkovski B, Serafimova N, Achkoski J. Novel approach for finding shortest route using Dijkstra's algorithm and fuzzy logic in a wireless sensor network integrated in a forest fire detection system. Environ Eng Manag J. 2020;19(6):1007–16. doi:10.30638/eemj.2020.095.
5. Sutton RS, Barto AG. Reinforcement learning: an introduction. Cambridge, MA, USA: MIT Press; 1998.
6. Boyan J, Littman M. Packet routing in dynamically changing networks: a reinforcement learning approach. In: Proceedings of the 7th International Conference on Neural Information Processing Systems; 1993 Nov 29–Dec 2; Denver, CO, USA. p. 671–8.
7. Mao H, Alizadeh M, Menache I, Kandula S. Resource management with deep reinforcement learning. In: Proceedings of the 15th ACM Workshop on Hot Topics in Networks; 2016 Nov 9–10; Atlanta, GA, USA. p. 50–6.

8. Kim T, Vecchietti LF, Choi K, Lee S, Har D. Machine learning for advanced wireless sensor networks: a review. IEEE Sens J. 2020;21(11):12379–97. doi:10.1109/jsen.2020.3035846.
9. Alsheikh MA, Lin S, Niyato D, Tan HP. Machine learning in wireless sensor networks: algorithms, strategies, and applications. IEEE Commun Surv Tutor. 2014;16(4):1996–2018. doi:10.1109/comst.2014.2320099.
10. Yang J, Li W, Li C, Zhang L, Liu L. An energy-efficient and transmission-efficient adaptive routing algorithm using deep reinforcement learning for wireless sensor networks. IEEE Internet Things J. 2025;12(23):50414–26. doi:10.1109/jiot.2025.3609624.
11. Okine AA, Adam N, Naeem F, Kaddoum G. Multi-agent deep reinforcement learning for packet routing in tactical mobile sensor networks. IEEE Trans Netw Service Manag. 2024;21(2):2155–69. doi:10.1109/tnsm.2024.3352014.
12. Akyildiz IF, Su W, Sankarasubramaniam Y, Cayirci E. Wireless sensor networks: a survey. Comput Netw. 2002;38(4):393–422. doi:10.1016/s1389-1286(01)00302-4.
13. Al-Karaki JN, Kamal AE. Routing techniques in wireless sensor networks: a survey. IEEE Wirel Commun. 2004;11(6):6–28. doi:10.1109/mwc.2004.1368893.
14. Schurgers C, Srivastava MB. Energy efficient routing in wireless sensor networks. In: 2001 MILCOM Proceedings Communications for Network-Centric Operations: Creating the Information Force (Cat. No. 01CH37277). McLean, VA, USA: IEEE; 2001. Vol. 1, p. 357–61. doi:10.1109/MILCOM.2001.985819.
15. Godfrey D, Suh B, Lim BH, Lee KC, Kim KI. An energy-efficient routing protocol with reinforcement learning in software-defined wireless sensor networks. Sensors. 2023;23(20):8435. doi:10.3390/s23208435.
16. Zhao B, Zhao X. Deep reinforcement learning resource allocation in wireless sensor networks with energy harvesting and relay. IEEE Internet Things J. 2021;9(3):2330–45. doi:10.1109/jiot.2021.3094465.
17. Keum D, Ko YB. Trust-based intelligent routing protocol with Q-learning for mission-critical wireless sensor networks. Sensors. 2022;22(11):3975. doi:10.3390/s22113975.
18. Suresh SS, Prabhu V, Parthasarathy V, Senthilkumar G, Gundu V. Intelligent data routing strategy based on federated deep reinforcement learning for IOT-enabled wireless sensor networks. Meas: Sens. 2024;31(6):101012. doi:10.1016/j.measen.2023.101012.
19. Zhang C, Patras P, Haddadi H. Deep learning in mobile and wireless networking: a survey. IEEE Commun Surv Tutor. 2019;21(3):2224–87. doi:10.1109/comst.2019.2904897.
20. He Q, Wang Y, Wang X, Xu W, Li F, Yang K, et al. Routing optimization with deep reinforcement learning in knowledge defined networking. EEE Trans Mob Comput. 2023;23(2):1444–55. doi:10.1109/tmc.2023.3235446.
21. Miuccio L, Riolo S, Samarakoon S, Bennis M, Panno D. On learning generalized wireless MAC communication protocols via a feasible multi-agent reinforcement learning framework. IEEE Trans Mach Learn Commun Netw. 2024;2:298–317. doi:10.1109/tmlcn.2024.3368367.