IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 5 September 2022, accepted 23 September 2022, date of publication 30 September 2022,
date of current version 14 October 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3210993

RESEARCH ARTICLE

Adaptive Routing in Wireless Mesh Networks Using Hybrid Reinforcement Learning Algorithm

SMITA MAHAJAN[1], (Member, IEEE), R. HARIKRISHNAN[2], (Senior Member, IEEE), AND KETAN KOTECHA[3]

[1]Department of Computer Science Engineering, Symbiosis Institute of Technology, Symbiosis International (Deemed) University, Pune, Maharashtra 412115, India
[2]Department of Electronics and Telecommunication Engineering, Symbiosis Institute of Technology, Symbiosis International (Deemed) University, Pune, Maharashtra 412115, India
[3]Department of Symbiosis Centre for Applied AI (SCAAI), Symbiosis Institute of Technology, Symbiosis International (Deemed) University, Pune, Maharashtra 412115, India

Corresponding author: R. Harikrishnan (dr.rhareish@gmail.com)

ABSTRACT Wireless mesh networks are popular due to their adaptability, easy-setup, flexibility, cost, and transmission time-reductions. The routing algorithm plays a vital role in transferring the data between the nodes. The network's performance is significantly impacted by the route opted by the algorithm. The router takes the decision to send the packet to the next router as per the policy of that algorithm. So even though that decision does not favor the right path selection, the router tends to follow its policy. This can be avoided by having intelligent routers that can make routing decisions on the fly. This paper presents the QL-Feed Forward routing algorithm (QFFR), a new generation of routing algorithms that combines reinforcement learning based on the Q-learning algorithm with a Feed Forward neural network. This algorithm (QFFR) can learn from the network environment and make routing decisions based on the algorithm's learnings. The AI agent's ability to select the fastest path, which enhances the efficiency of the routing operation, is demonstrated by the working of the suggested QFFR algorithm. This paper also evaluates the performance of traditional algorithms, namely, Ad-hoc On-Demand Distance-Vector, Optimized-Link-State-routing, Destination-Sequenced Distance-Vector and Distance Source routing. The evaluation parameters include throughput, packet delivery ratio, and delay. The parameters are the outcomes of the time the information takes to reach from source to destination. This analysis highlights the improvement in the routing decision ability of a router. As per analysis, Ad hoc On-Demand Distance Vector Algorithm outperforms with throughput 723.13 Kbps, delay 343.73 ns. Q-learning agent identifies the route and reaches the destination in average of 3.7s in non-grid architecture. The Q-learning agent takes 0.49sec with a grid size ten by ten and 0.53sec in three by four grid size. The suggested QFFR takes 7.62s score-over time with stable, consistent performance.

INDEX TERMS Deep Learning, reinforcement learning, Q-learning, Markov decision process, Routing Algorithms, Wireless Mesh Networks.

I. INTRODUCTION

During this era of digital development, networks play a critical role. Figure 1 shows Wireless Mesh Networks with a mesh architecture. Wireless mesh networks' key advantages are their versatility and customizing capabilities. Any future modifications would be simple to accommodate, resulting in lower expenses and upkeep of the network. Wireless mesh networking is a recent development that emerged from a decade of Ad-hoc networking development [1]. A wireless mesh network (WMN) is a decentralized network system built on existing wireless technology, namely the 802.11 standards, that works on an Ad-hoc communication mode [2], [3]. The wireless mesh network is ideal for next-generation communication with its flexibility and extensive coverage. Multi-radio mesh routers and single-radio mesh

The associate editor coordinating the review of this manuscript and approving it for publication was Chan Hwang See.

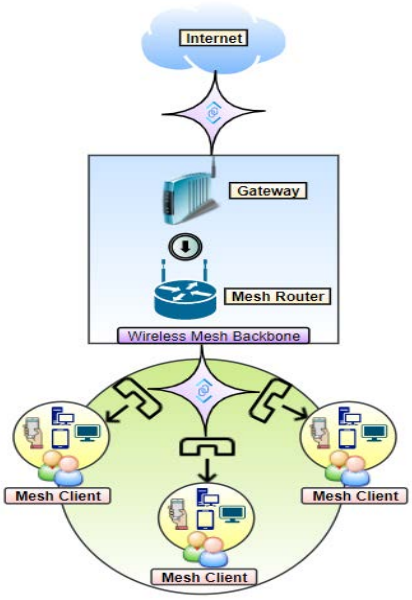


FIGURE 1. Wireless mesh network.

clients form Wireless Mesh Networks(WMN) [4]. As a result, such networks are developed by connecting wireless access points at each network client's location [5], [6]. Mesh nodes are radio transmitters that are modest in size. They behave in the same way as a wireless router does. The software deployed on these nodes guarantees that they communicate with one another through the network and governs how they interact [7]. The nodes determine the fastest and safest path in a dynamic routing mechanism. Mesh routers (MRs), Mesh clients (MCs), and gateways are integral parts of the wireless mesh network [8]. The wireless mesh backbone is formed by MRs, enabling multi-hop connection for mobile MCs, enabling end-users to communicate with others or the Internet via access points (APs) [6], [8]. In the current scenario, Internet application services, such as Voice-over-IP (VoIP), video streaming, audio streaming, and online games, are in high demand. Because of the COVID spread, there has been significant growth in online transactions in the last two years [9], [10]. Multimedia services account for more than half of all current Internet traffic. As a result, network providers must undergo monitoring and maintenance duties to guarantee that end-user Quality-of-Experience is adequate [11]. The routing algorithm is one of the most important aspects directly impacting the network's performance. On the other hand, traditional routing algorithms do not contemplate the network data history, such as equipment failures and overcrowded paths, which have affected the earlier communication. Routing techniques based on machine learning intend to benefit when leveraging network data. With people's rising need for communication, technologies and processes must be constantly improved to guarantee the user's experience [12], [13]. Traditional Ad-hoc network routing algorithms, such as Optimized Link State Routing (OLSR), Ad-hoc On-Demand Distance Vector (AODV), and others, cannot know or learn from suspicious network incidences or events that have taken place various times in the past. As a result, these algorithms can opt for a route with typical faults in the past [11]. Routing strategies that adapt to changes in traffic trends, traffic load levels, and network topologies are required to route information packets in continuously varying communication networks effectively. These shortest path-based approaches provide reduced network latency when the amount of data is small. However, many routers selected by several pathways may experience severe traffic congestion when the volume of network data traffic multiplies. The network will become congested when the data volume exceeds the buffer capacity of the chosen routers, reducing network throughput and increasing network delay.To put it another way, conventional routing protocols cannot adapt their transmission techniques in response to changing network conditions [14], [15]. Machine Learning algorithms have been working in diverse applications in recent years. Similarly, Machine Learning algorithms can effectively route control protocols since Reinforcement Learning (RL) is increasingly used to tackle many complex challenges [16]. In Reinforcement Learning, an agent must be able to learn how to act in a dynamic environment iteratively. For example, an agent who makes a decision is rewarded or punished depending on whether the decision was Good or No [17]. The Reinforcement Learning (RL) approach can improve the nodes' ability in the decision-making process of route choice, resulting in increased network performance. As a result, improved metrics such as improved throughput decreased delay and, eventually, the application services and user experience [14]. The fundamental idea behind this method is to decide whether or not to transfer a packet with a specific destination address to an intermediate node based on the current estimation values of a group of neighbours. The estimation values are dynamically updated by rewards, which are feedback values for the chosen action - a packet forward event to a particular neighbour [12], [18]. According to the literature, the traditional algorithms which are widespread in use, it is evident that each of them has some merits in specific situations. For example, OLSR works better than AODV in all aspects except delay when there are more than fifty nodes in a mobile network. Even though OLSR performs better in mobile conditions than AODV and DSDV, AODV possesses high throughput in static conditions [19]. However, under some circumstances, AODV exhibits better latency than DSDV, DSR, and OLSR [20]. The suggested algorithm uses well-known artificial intelligence techniques. So, it becomes an essential task to measure its performance against the traditional benchmarked algorithms. Based on assessment metrics such as packet loss, receiving rate, delay, and throughput, this paper appraises the performance of traditional techniques, namely, Optimized Link State Routing (OLSR), Ad hoc On-Demand Distance Vector (AODV), Destination-Sequenced Distance-Vector (DSDV) and Distance Source Routing (DSR). This complete analysis shows that RL-based

algorithms can be applied as an Adaptive Strategy for enhancing network performance. This paper presents the QL-Feed Forward Routing algorithm (QFFR), a millennial age routing algorithm that combines Q-learning, one of the efficient reinforcement learning algorithms, and a Feed-Forward Neural Network, again a very well-known deep learning network. The suggested QL-Feed forward Routing algorithm (QFFR) model has a continuous auto-learning capability from the network environment with which it interacts and can make routing decisions based on the AI agent's gained experience. This paper includes an environment produced by a three-by-four grid. This grid architecture is synonymous with the ''Taxi-V3'' environment from the well-known ''open AI Gym''. This suggested algorithm's performance is also evaluated in a network environment built by producing a random mesh of nodes to simulate a real-world situation. Both of these approaches ensure the novelty in implementing Reinforcement learning to identify the best route faster. Young researchers in this field would also benefit from this paper, shedding light on the modern reinforcement learning techniques. Additionally, very little noteworthy research or analysis was conducted for fast determining the shortest path, which improves the performance of the Wireless Mesh Networks. All of these emphasize the novelty and room for developing the suggested algorithm.

The following are the significant contributions made by this paper:

1) Q-learning Feed forward Routing algorithm (QFFR), a hybrid algorithm, is suggested for selecting the best path to send the information from source to destination for a Wireless Mesh Network. This algorithm combines Q-learning, one of the efficient reinforcement learning algorithms, and a Feed forward neural network, a well-known deep learning network.
2) Based on performance parameters such as packet loss, receiving rate, delay, and throughput, this paper appraises the performance of traditional algorithms, namely, Ad hoc On-Demand Distance Vector (AODV), Optimized Link State Routing (OLSR), Destination-Sequenced Distance-Vector(DSDV) and Distance Source Routing(DSR).
3) This paper compares the performance of the suggested QL-Feed forward Routing algorithm (QFFR)s with the well-known traditional algorithms, as well as the Reinforcement Learning algorithm Q-learning applied for selecting the route in a wireless mesh network.
4) This paper describes the best ways for tuning the important hyper-parameters of reinforcement learning algorithms (e.g., learning rates, discount factor) for improved self-learning and interacting with the underlying network environment for more optimum routing decisions.

The following is the outline for the paper: An overview of previous work in this field is presented in Section 2. The conventional routing algorithms are described in Section 3. Section 4 delves into the machine learning-based state-of-the-art analysis methodologies used for routing. Section 5 presents and analyzes the obtained results, while Section 6 shows the conclusion of the findings.

II. OVERVIEW OF RELATED WORK

Routing is a fundamental component of network deployment to maintain and enhance system performance. Given the rapid requirements among several wireless applications, accuracy(lossless), latency, and effectiveness are significant challenges in next-generation communication [1]. According to Ruijin Ding *et al.*, conventional routing protocols such as OSPF, IS-IS, RIP, and EIGRP are becoming increasingly inadequate for networks with large amounts of data, high data rates, and low latency requirements [14]. The implied reason is that these protocols calculate the shortest path from a source router to its destination without considering actual network variables like each router's remaining buffer size or failed router's current status [14]. The nodes or links in wireless mesh networks vary over time, and each wireless link between two nodes is established or broken regularly. This challenge is based on network conditions, which mainly include the variation in the received signal intensity while on the journey to the destination. This situation worsens when a significant distance disperses nodes. A wireless node powered by a battery will fail when the battery runs out, or the node overheats and shuts down. As a result, the network's topology constantly changes, making routing more difficult. [3]. Information is sent to distant nodes in communication networks in packets. Routing strategies are vital in delivering these packets to their intended destinations to decrease delay and reduce the congestion occurring due to increased numbers of packets travelling on the network. Routing strategies should be able to respond to changes in network conditions, which are the results of traffic load, traffic patterns, and network topology [21]. Wireless Mesh Networks are becoming increasingly prevalent in current situations due to their infrastructure and well-organized design. An Ad-hoc network incorporates a group of nodes that connect without a centralized infrastructure. This kind of network can self-organize and reconfigure itself when a node connects or departs from the network. Fixed and movable nodes are the two types of nodes involved in wireless mesh networks [22]. Although ad hoc networks do not require a precise structure, they need a standard communication and management method. A convention, such as an ad hoc routing protocol, is necessary for mobile ad hoc network nodes to decide the route the packets must travel to reach the specified destination. Furthermore, because the nodes in an ad hoc network are oblivious of each other initially, they must discover each other by broadcasting their presence to neighbouring nodes [21]. AS per Sidoine D *et al.*, when the basic routing algorithms Ad hoc On-Demand Distance Vector (AODV), Optimized Link State routing(OLSR), Destination-Sequenced Distance-Vector (DSDV) and Distance Source Routing (DSR) were compared on three metrics: packet loss, routing overhead, and

route length, DSR and AODV beat DSDV [22], [23]. When comparing the energy consumption of the Optimized Link State Routing Protocol (OLSR) with DSR, it was observed that DSR uses its routing policy when the traffic rate is low. However, OLSR performs better when the traffic rate is higher [22]. In particular, conventional routing protocols such as OSPF, IS-IS, RIP, and EIGRP are becoming increasingly inadequate for networks with large amounts of data, high data rates, and low latency requirements. The fundamental reason is that these protocols compute the shortest path from a source router to its destination without assessing actual network variables like each router's remaining buffer size [14]. Machine Learning is a hot technology that has succeeded in diverse fields, including image processing, text mining, natural language processing, and agent systems. Because of its intelligence, ML is also adaptable to network architecture for WMN. Understanding and predicting network conditions are necessary and helpful, and ML can do it efficiently and accurately [12]. Machine learning has recently found widespread use in wireless communications routing tasks. [24]. Q-learning, a model-free method, has been recognized as a promising approach, mainly when applied to intricate decision-making procedures. Q-learning offers several benefits, including excellent learning capabilities, effective results, and the capacity to integrate with other models s [25]. As per Rujin Ding *et al.*, the fifth generation (5G) of cellular mobile communications is characterized by high data rates, ultra-low latency, high energy efficiency, and widespread device connectivity [14]. Qion Gxiao fu *et al.* stated that the ideal selection policy is centralized and necessitates local channel state information (CSI) for all hops, resulting in considerable computing complexity and signalling overhead. They have proposed modeling the multi-hop relay selection problem as a Markov decision process (MDP) and solving it by applying a decentralized Q-learning approach [24]. Khamxay Leevangtou *et al.* suggested a reinforcement learning approach to improve the controller performance in software-defined wireless mesh networks (SDWMN) for dynamic changes in network topology with link or node failures [3]. Using Deep Q-learning and the Convolution Neural Network (CNN)method, Hyansu Bae *et al.* suggested a noble multi-robot path planning system. According to them, Deep Q-learning strengthens the learning algorithm and is integrated with the CNN algorithm to study the environment [26]. According to Duonga *et al.*, wireless mesh networks are becoming more commonly used as traffic demand grows. Due to this, traditional routing systems face a substantial barrier in monitoring and evaluating Service. This paper presents a QoS-assured intelligent routing system for WMNs with high traffic loads that use reinforcement learning to improve performance [27]. As per Saeed Kaviani *et al.*, DeepCQ+ routing, which combines emerging multi-agent deep reinforcement learning (MADRL) in a novel way, achieves persistently higher performance across a wide range of MANET architectures while training only on a limited range of network parameters and conditions [28]. In a fully decentralized environment, Xinyu You *et al.* developed a unique packet routing framework based on multi-agent deep reinforcement learning. In this framework, each router has its own independent long short-term memory (LSTM), a recurrent neural network(RNN) for training and decision making [29]. A Case of VoIP Service Using Enhanced Routing Algorithm Based on Reinforcement Machine Learning is introduced by Davi Ribeiro Militani *et al.* Using energy as the significant selection parameter [11]. Zahoor Ali Khan *et al.* demonstrated the performance of a Q-learning-based energy-efficient routing algorithm for underwater acoustic sensor networks [30]. According to Gerard George *et al.*, an AI agent aims to optimize routing decisions. The Route Bricks architecture is a revolutionary network design that processes packets in software running on a cluster of general-purpose PC hardware [10]. Due to their inability to adjust to the overwhelming dynamics of underwater settings, routing systems for multi-hop underwater wireless sensor networks suffer significantly from performance loss. Valerio Di Valerio *et al.* provide a brand-new data forwarding strategy where relay selection quickly adjusts to the changing underwater channel conditions to address this issue. The proposed protocol 'CARMA', for Channel-aware Reinforcement learning based Multi-path Adaptive routing. They suggested adaptively switching between single-path and multi-path routing under the path of a distributed reinforcement learning framework that simultaneously optimizes route-long energy consumption and packet delivery ratio [31]. Recently, energy-based systems have substantially improved because of machine learning and wireless sensor networks (WSN) integration. However, some challenging issues need innovative analytic approaches with low energy use. Additionally, the environment in which WSN operations occur is risky and impossible to forecast. Multiple network threats can compromise the accuracy and security of data collecting. Therefore, protecting low-power sensors from these threats should be a primary focus. Saba *et al.* have thus presented an autonomous IoT Security strategy based on machine learning to achieve the best feasible levels of energy economy and dependable transmissions [32]. A unique multi-hop multi-frequency mesh design based on Software Defined Networking (SDN) has been introduced by Abdollahi *et al.* Here, Off-the-shelf WiFi chips are modified and integrated into a novel, light hardware architecture to enable Adaptive routing and frequency selection. The well-known Dijkstra algorithm has been enlarged to fit the new multi-hop multi-frequency platform [33]. A robust and reliable system for IoT -based mobile mesh networks, which ensures reliable routing, data confidentiality, and integrity (RTS), was put forth by K. Habib *et al.*. Based on wireless channel measurement and network parameters, the suggested method enables dependable data routing between mobile mesh clients, routers, and gateway devices [34].

It is worth noting that work on decision-making for selecting a route to destination intelligently in wireless mesh networks has rarely focused on intelligent traffic routing, which can be employed to enhance network performance.

Furthermore, a standard publication presenting all existing machine learning and deep learning approaches could benefit budding investigators. Again, there is little prominent work applying routing strategies to modern trends, particularly Artificial Intelligence tools and methodologies. These aspects emphasize the need for this research effort and the room for development.

III. ROUTING ALGORITHMS IN WIRELESS MESH NETWORKS

Routing is a fundamental component of networking deployment to keep and enhance system performance [8]. Packet loss, receiving rate, delay, and throughput are essential elements in determining the network's quality of Service (QoS). The network and MAC layers are responsible for identifying the route for which the routing protocols are used. As per the routing algorithms' strategy, the routes are identified, over which the packets transverse towards the destination [35]. Routing Protocols explore the routes for packet transmission and deliver packets to a source's desired destination. Routing algorithms are used to find the best path for data communication between nodes in a network. Proactive Protocols (Table Driven), Reactive Protocols (On-Demand), and Hybrid Protocols are the three types of protocols [36].

A. AD HOC ON-DEMAND DISTANCE VECTOR(AODV)

Reactive routing algorithms wait for demand before finding a route to a packet's destination. AODV is by far the most widespread reactive routing protocol. This protocol provides the foundation for several reactive routing algorithms. The network architecture constantly fluctuates since nodes are movable; AODV leases nodes to receive routes to new destinations quickly [7]. During this procedure, a packet containing a route request RREQ is flooded from the source node. Each node that gets this packet passes it on to other nodes till the packet reaches its destination [37]. During the first stage, all intermediate nodes contemplate the route to the source as per the RREQ packet's information about the route to the source. This node delivers a route reply RREP packet when it reaches the destination. This packet follows the RREQ's course in reverse. RREP stipulates a path to the destination to all transitional nodes on the way back to the source. When RREP reaches the source, the discovery procedure is complete. The packet transmission process can now begin. Each intermediary node knows to which neighbour it should deliver the packets to reach the source or destination [38], [39].

B. OPTIMIZED LINK STATE ROUTING (OLSR)

The proactive routing protocols do not wait for a request to find a route to a destination; instead, they maintain a table for this purpose [40]. So, they are recognized as table-driven routing protocols. Maintaining a routing table at each node establishes the route in advance for data transfer [41]. OLSR applies the mechanisms that include link sensing for examining the connectivity of nodes. It is performed by sending periodic HELLO messages via the interfaces used to check connectivity. For each interface,

1) A special HELLO message is generated.
2) Neighbor detection: In a network with single interface nodes, the set of neighbour nodes can be inferred from the information transmitted as part of link sensing. One of a node's single interfaces serves as its address. The number of interfaces per node affects this process of node detection.
3) MPR Selection and MPR Signaling: Each node chooses a group of its neighbours to act as multi-point relays (MPRs). Only those MPRs will re-transmit broadcast messages to receive them by all nodes two hops away.
4) Topology Control Message Diffusion: Topology control is used to build the routing table at each node using Topology Control (TC) packets, which MPR exclusively carries.
5) Route Calculation: For route calculation, each node's routing table, which has adequate link-action information, will be employed [42].

C. DESTINATION-SEQUENCED DISTANCE VECTOR ROUTING (DSDV)

Destination-Sequenced Distance Vector Routing (DSDV) is a table-driven routing algorithm for ad hoc mobile networks based on the Bellman-Ford algorithm. In this algorithm, a sequence number is designated to each entry in the routing table, usually, an even number if a connection is established; otherwise, an odd number is assigned [43]. The destination generates a number, which the emitter must include in the next update. Complete dumps are delivered infrequently to send routing information between nodes, but minor incremental changes are sent more frequently. The most recent sequence number is used when a router receives new data. The route with the best metric is selected if the sequence number matches one already in the table. Entries in the table that have not been changed in a long time are considered expired or stale. Such entries are removed, as these routes would otherwise use those nodes as future hop. Such entries and routes that leverage those nodes as future hops are eliminated [44].

D. DYNAMIC SOURCE ROUTING(DSR)

The source routing strategy is used in this protocol, which means that almost all path information is stored (and updated) at mobile nodes. The only two critical processes are Route Discovery and Route Maintenance. A Route Reply RREP is generated only if the message has reached its intended destination node (route record which is initially contained in Route Request would be inserted into the Route Reply RREP) [45]. To return the Route Reply, RREP, the destination node must have a route to the source node. If the route is in the destination node's route cache, that route will be used. Else, the node will reverse the route depending on the route record in the Route Request RREQ message header [46].

TABLE 1. Summary of traditional algorithms.

Ref	Name	Type	Description
[47]	AODV	Reactive	1) It's routing method is hop-to-hop 2) If it is necessary to understand the path to a given destination, the node creates a Route Request (RREQ). 3) When the node receives the request it generates a Route Reply (RREP) which contains the number of hops required to reach the destination.
[11]	OLSR	Proactive	1) Packet s can and do go out the same interface in wireless ad hoc networks, necessitating a different technique to optimize the flooding process 2) The OLSR protocol discovers two hop neighbour information at each node and elects a set of multi point relays in a distributed manner.
[44]	DSDV	Proactive	1) It is a Wireless Ad hoc network protocol 2) A Unique Sequence number per broadcast is assigned 3) The main contribution of the algorithm was to solve the routing loop problem.
[45]	DSR	Reactive	1) Operate entirely on-demand basis. 2) Source routing is a routing technique in which the sender of a packet determines the complete sequence of nodes through which the packets are forwarded. 3) DSR uses a reactive approach in which there is no need to periodically flood the network with table update messages which are required in a table driven approach 4) The intermediate nodes also efficiently use the route cache information to reduce the control overhead

All these algorithms have some specific characteristics or features.Table 1 provides an overview of the aforementioned algorithms.

E. REINFORCEMENT LEARNING BASED ROUTING PROTOCOLS

Reinforcement learning is training machine learning models to make a series of decisions in a specific order. In an uncertain, potentially complex environment, the agent learns to achieve a goal [48]. The working of RL interaction is depicted in Figure 2, where an agent has a set of actions' A' to choose from and interact with the given environment. The environment reacts to the agent's action in the form of a reward, reinforcing the agent with more knowledge about itself. The agent subsequently uses this new information to alter the future selection of actions in response to environmental conditions, which is usually regulated by the introduced estimation value Q [48]. The RL optimization process may be described theoretically as a Markov Decision Process (MDP), which introduces four sets: S, A, P, and R [50] where,

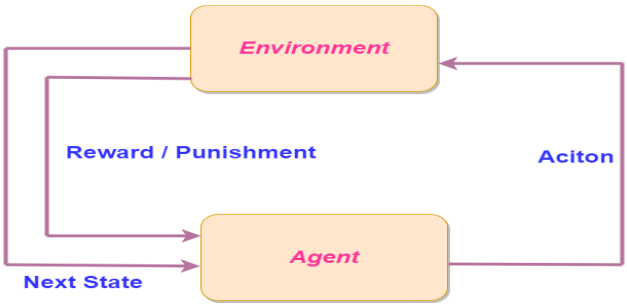


FIGURE 2. Working of reinforcement learning [49].

S – is a set of States that an agent can be in at any particular time

A – is a set of conceivable actions that an agent can take at any particular time

P – is a set of possibilities that an agent in action 'S' would transit to State S' in t+1' time by executing action A

R - Reward received by the agent as a result of taking some specific action and changing its State from S to S' As per MDP's logic, the probability of transitioning from s to s' after executing an action is stated as given below:

$$P^{a}_{ss'} = P_r(S_{t-1} = S' | S_t = S, a_t = a) \tag{1}$$

In equation (1), the corresponding estimation reward values for a given action a, from State' s' to 's', are represented as shown: The sets S and P are viewed in terms of RL theory as a single set of estimation Q-values, which is reliant on the feedback reward value from the environment, as well as the current time t when the relevant action was completed [51]. The equation below is a representation of the RL estimating values function.

$$R^{a}_{ss'} = Er_{t+1} | S_t = S, a_t = a, S_{t+1} = S' \tag{2}$$

$$Q_{k+1} = K + alpha * [r_{k+1} - Q_k] \tag{3}$$

Q_k is the reward estimation value from the previous step, Q_{k+1} is the reward estimation value from the current step, and r_{k+1} is the reward value for a current step action. The alpha *alpha* is a learning parameter for step size; and K is the current step number.

F. WORKING OF Q-LEARNING

Q-learning is a form of Reinforcement Learning in which AI agents operate in a controlled environment, with States and rewards as inputs and actions as outputs. Model-free environments are used in Q-learning. In a model-free environment, the AI agent tries to build an optimal strategy by interacting with the environment directly. The AI agent accomplishes this by employing an explicit trial-and-error strategy, in which it repeatedly tries to solve the problem using various ways across multiple episodes while continuously updating its policy as it learns more about its surroundings [52]. Q-learning models have input and output system rewards in an environment. The agent's action is an input to get the reward(output) from the environment with which it interacts. They use the Markov Decision Process and have two modes: training and inference. Q-learning models have two unique properties in addition to these essential characteristics. First, in the Q-learning model, the number of alternative States is finite. The AI agent will always be in one of a predetermined number of scenarios. Second, the number of actions that can be taken in Q-learning is also finite. The AI agent will always have to pick between a specific set of options for what to do next. By relying on and updating Q-values, the AI agent learns about the surroundings [53]. Q-value indicates the quality of a particular action 'a' in a given State S:(S, a), which can be represented as a function of Q with two input parameters, S and a. An AI agent always looks for the highest quality action in any state. Q-values are current estimates of the sum of future rewards. That means Q-values estimate how much additional reward can be accumulated through the remaining steps in a current episode if an AI agent is in State S and decides to take an action a. The Q-values, therefore, increase as the AI agent gets closer and closer to the highest reward. The rewards can be positive or negative. A negative reward can be conceptualized as a punishment or penalty. The objective of an AI agent is to maximize its total rewards and minimize the negative rewards(punishment). The Q-values are stored in a Q-table with one row for each possible State, and one column for each possible action [11], [54]. An optimal Q-table contains values that allow the AI agent to take the best action in any possible State, thus providing the agent with the potential path to the highest reward. The Q-table, therefore, represents the AI agent's policy for acting in the current environment [28]. In Q-learning, the Temporal Differences' TD' provide a method of calculating how much the Q-value for the action taken in the previous State should be changed based on what an AI agent has learned about the Q-values for the current State's actions. Figure 3 shows the formula for calculating the temporal difference. This means that in the Q-learning process, the Q-value for the most recent action is always updated after each step. Thus, if the new State is promising, that is, if it provides a relatively good reward, then the Q-value for the previous action will be increased, and the AI agent will rely on this knowledge to make the decision during the next training episode [13]. The Bellman

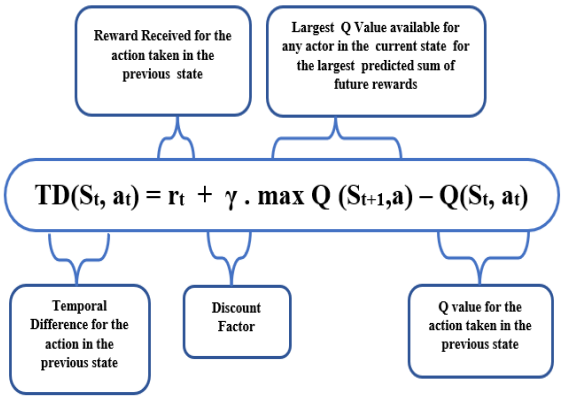


FIGURE 3. Temporal difference [13].

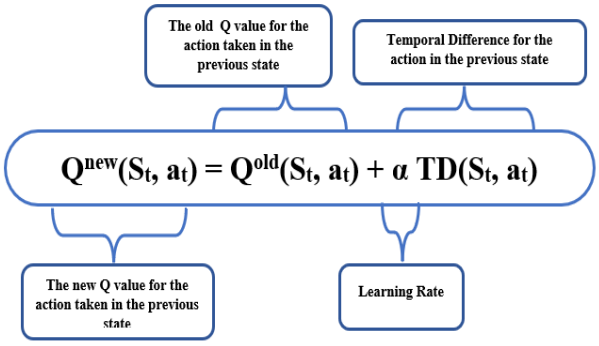


FIGURE 4. Q-Value [55].

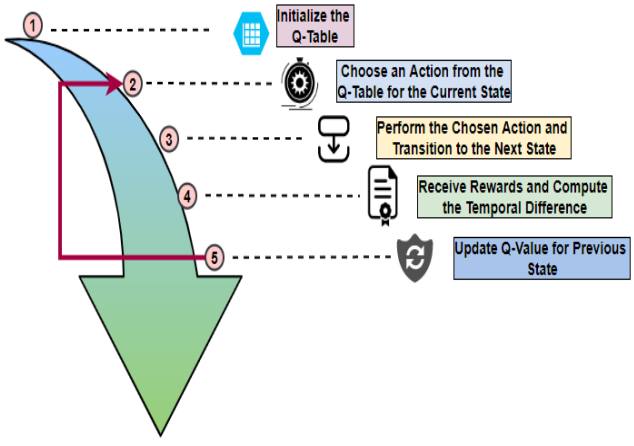


FIGURE 5. Q learning.

equation, invented by Richard Bellman, is used to calculate what new value is to be used as the Q-value for the action taken in the previous State, as shown in Figure 4. It relies on the old Q-value for the action taken in the previous State and what has been learned after moving to the next State. It includes a learning rate parameter alpha (*alpha*) that defines how quickly the Q-values are adjusted. Figure 5 shows the Q-learning procedures as a whole.

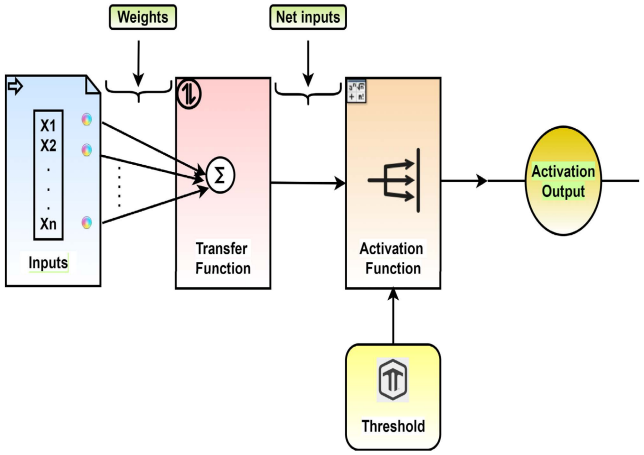


FIGURE 6. Mathematical formalism of a neuron.

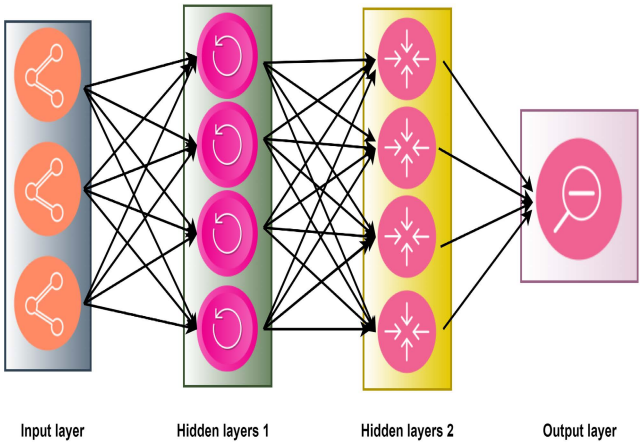


FIGURE 7. Feed Forward network.

G. FEED FORWARD NEURAL NETWORKS

Because the input is only processed in one direction, the Feed Forward model is the most basic form of a Neural Network. The data flow in only one direction and never backwards, regardless of how many hidden nodes it passes through [25]. A feed-forward Neural Network is a biologically inspired classification technique. It is stacked throughout and comprises a (relatively vast) number of basic neuron-like processing elements. Every unit in a layer is connected to every unit in the layer preceding it [53]. These connections aren't designed to be all the same: their strength and weight may vary. The weights represent the knowledge of a network on these links. The nodes are the basic building blocks of a Neural Network. Data enters through the inputs and passes through the network layer by layer until it reaches the outputs. During normal operation, there is no feedback between layers when it serves as a classifier. It's for this reason that they're known as Feed-Forward Neural Networks. In a mathematical formalism, learning entails adjusting the weight coefficients to meet certain conditions. A typical neuron has a linear activator and a non-linear inhibitory function as shown in Figure 6. The sums of weighted inputs plus an independent term bias, b, are produced using the linear activation function. The sum's signal level is attempted and seized using the non-linear inhibitory function [56]. The most prevalent inhibitory functions are Step, Sigmoid, and Hyperbolic Tangent functions as shown in Figure 7. Purely linear functions are also employed for this, especially in output layers. Q-learning has been proposed to create routing policies for path selection. The Q- routing algorithm requires nodes to make per-node routing decisions. Each node learns a local deterministic routing policy using the Q-learning technique. Finding a local routing solution requires less time to compute than finding a global routing solution. For each State-action pair, the algorithm keeps a Q-value Q(s, a) in a Q-table. Let 's_t' and a_t signify an agent's current State and action at time instant t, respectively. Furthermore, let r_{t+1} signify the reinforcement signal generated by the environment for taking action in the State S_t. When the agent obtains the r_{t+1} reward, it updates the Q-value for the State s_t and action a_t as follows:

$$Q(S_t, a_t) = Q(S_t, a_t) + \alpha * [r_{t+1} + \gamma * Max(Qa_{t+1}(QS_{t+1}, a_{t+1}) - Q(S_t, a_t))] \quad (4)$$

where learning rate alpha (*alpha*) [ranging between 0 to 1] is the learning rate, and discount factor gamma (*gamma*) [ranging between 0 to 1] is the discount factor. Each node of the switching network contains Q learning modules. Local information is transmitted between nodes regularly to maintain track of projected delivery times to nodes. Routing policies contribute to discovering paths that require the least amount of delivery time. The overall time it takes to send a packet is used to evaluate the policy's performance. Each node calculates the time required to travel to a destination from every neighbouring node. A table is maintained at each node to do this. The table's value tells how long it will take for a packet P to reach destination d if it is sent through neighbour y. When node x receives a request to transfer a packet to destination d, it delivers it to the neighbour y' with the least estimated delivery time to 'd'. Following that, 'x' modifies its policy and questions 'y' how long it will take 'y'' to deliver the packet to 'd'. Because 'y''s estimate is more likely to be closer to 'd', it's used to update 'x' s delivery time estimate [29], [48], [57]. On their journey to their destination, packets usually make several hops at intermediate nodes. Each node receives a packet, stores it, and then forwards it to the next hop until it reaches its destination. Q-routing is an Adaptive routing method that routes packets based on information learned from its neighbours [57].

H. PROPOSED QL-FEED FORWARD ROUTING ALGORITHM (QFFR)

Using the Q-value iteration, the Q-table containing the Q-values of any State-action combination is generated using

the Q-learning algorithm. Because every State-action pair is stored, the Q-learning algorithm works effectively for finite States and actions spaces because it requires a large amount of memory to store all of the State-action pairs and many more iterations for the Q-table to converge. It is simply impossible to utilize the Q-learning method when the States space, actions space, or both are very large (in a real-life scenario). The Q-value is estimated using Deep Neural Networks, known for their effectiveness in approximating functions, as a solution to this problem. The Q-table can be viewed as a function used to evaluate an unknown function at specific locations. Since it is a function, Deep Neural Networks may be used to approximate it. The RL process employing Q-learning and feed-forward network s is depicted in Figure 5. The AI agent is assigned the role of a node in this proposal, which is a Wireless Mesh network node. For delivering the packet to the next router, the agent must do the six steps: up, down, left, right, pick up the packet, and deliver the packet. The set of rewards that the agent receives depends on ACK or NACK, which is represented as a routing table of each node. Q-learning and a Feed Forward Network are combined in the proposed QL-Feed Forward routing algorithm (QFFR). The States of the environment are the inputs to the Feed forward neural network, and the outputs are the collection of Q-values for each action connected with the input State. Thus the Neural Network s are employed to estimate the Q-value rather than the Bellman equation. A Neural Network learns by repeatedly considering a set of input values and attempting to predict their output values as correctly as possible. This is performed by updating the weights for all nodes in the input layer and hidden layer(s) iteratively to reduce overall prediction error. The weights are significant in establishing each artificial neuron's input and output values and forecasting the outcome. The AI agent will usually take action with the highest Q-value, just like Q-learning. The targets of an input State in deep Q-learning are the Q-values associated with each conceivable action for that input State. The present forecasts of the sum of all future rewards earned corresponding to each action are known as Q-values. The temporal difference formula calculates these Q-values equation as shown in Figure 3. The temporal difference is a formula used to find the Q-value by using the Q-value of the current State and action and that of the previous State and action. The SoftMax function is used in the QL-Feed Forward routing algorithm (QFFR) to achieve exploration. The AI agent must consider all of its alternatives to determine how each will affect the agent's goal of maximizing overall rewards. The SoftMax function transforms a State's set of Q-values into a set of probabilities for each potential action in that State. The probability distribution for the Q-values is obtained using the Soft Max function. The AI agent's decision is then made by drawing a random number from the probability distribution supplied by the Soft Max function. As a result, the AI agent is more likely to take the action that appears to offer the best reward. The entities

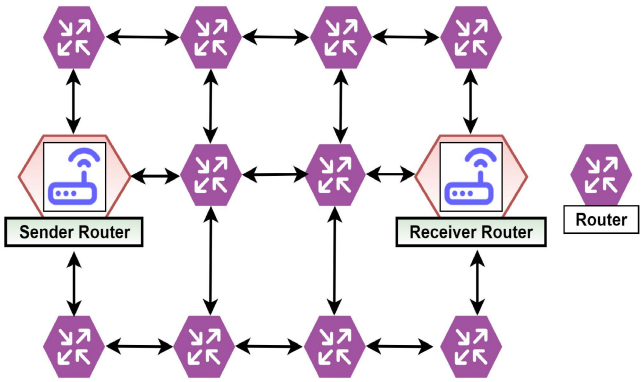


FIGURE 8. Routers in grid architecture.

involved in Reinforcement Learning and those of suggested routing protocols are related as stated below:

- Agent is acting as a Source node(Sender).
- The set of actions is the 'Set of nodes' that constitutes the network.
- Set of estimation Q-values forms the 'routing Table'.
- Agent's action is to send a packet to a neighbour.
- The agent is Rewarded when the node receives the Acknowledgment (ACK).

The set of actions or action space refers to the nodes which constitute the network on which sender node (router) R5, as shown in Figure 8, sends the messages. Sending a packet to a given network node is an action. The agent has to send the packet either in the up, down, left, or right direction as it is assumed that the architecture of the network is the grid formed by nodes at each cell position in the grid, as shown in Figure 8. Node R5 expects to receive an Acknowledgement Message (ACK) while transmitting this packet; if this happens, it indicates that the given node received the message, and a reward was earned for selecting this node to send the message. Node R5 expects an Acknowledgement Message (ACK) while transferring this packet; if this occurs, it implies the selected node received the message. A reward was won for choosing the above node to deliver the message. If the ACK is not received, it means the message was lost, and the route is poor; node R5 is therefore penalized for the action taken. Finally, the routing table of the protocol determines the best efficient way to transmit a packet to a particular destination. Similarly, an estimate set determines which activity yields the optimal reward. As shown in Figure 9, the Neural Network takes a State as input and returns the Q-values of all feasible actions for that State. The feed-forward network's input layer is the same size as a State, and the output layer is a collection of Q-values for each action associated with the input State. This strategy can handle various sophisticated decision-making processes that a machine with human-like intelligence traditionally could not. In this QL-Feed Forward routing algorithm (QFFR), the AI agent (WMN node) preserves all of its previous experiences as well

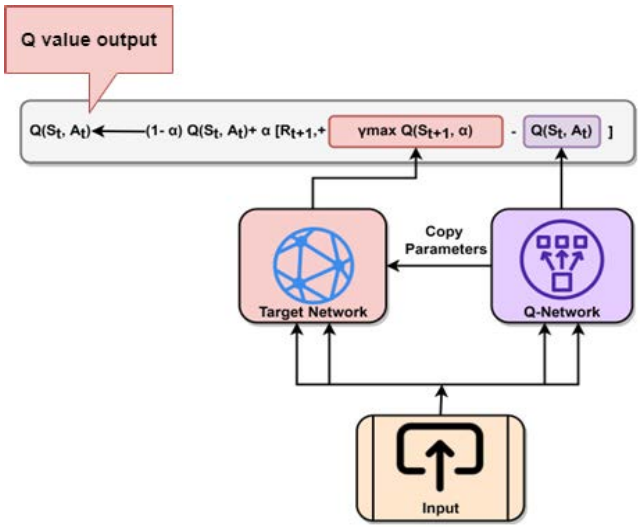


FIGURE 9. QL-Feed forward network.

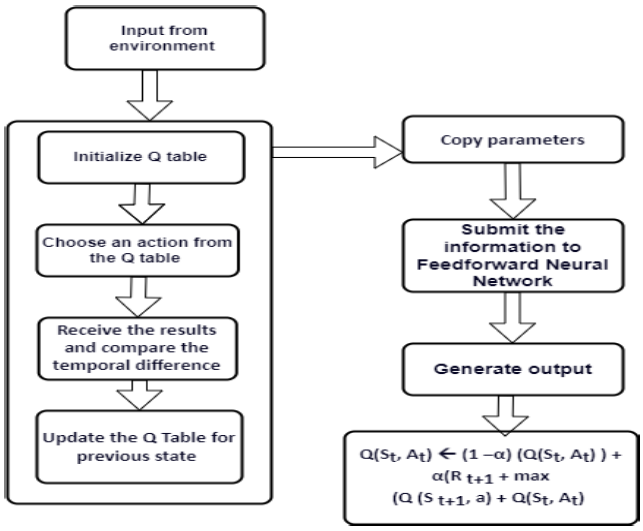


FIGURE 10. QL-Feed forward routing algorithm.

as the future action determined by the Q-output. To stabilize the training and block the abrupt rise in Q-value count, the Q- Network gains the Q-value at State' s_t'. At the same time, the target network (Neural Network) calculates the Q-value for State' S_t_+1 '(next State) by reproducing it as training data on each iterated Q-value of the Q-network. Figure 9 below depicts this procedure.

$$Loss = \gamma + Max(a'Q(S', a') - Q(S, a)^2) \quad (5)$$

Previous experiences must be gathered and stored by the AI agent. When a Neural Network and Q-learning are combined, the Neural Network stores the experience in memory in a tuple format (State, Next State, Action, Reward). It has been shown that selecting a random value from a batch of previous data enhances Neural Network training stability. So, to improve agent performance, the QL-Feed Forward routing algorithm (QFFR) leverages another concept: experience replay, which is simply the stocking (remembering) of previous experiences. The target network uses experience replay for training and by the Q- the network for calculating the Q-value. The squared difference between the targeted and forecasted Q-Values is used to compute the loss. This is solely done to train the Q-Network before copying the parameters to the target network. The working of QFFR can be summarized simplistically as given below:

1) Provide the agent with the current status of the environment.
2) The agent makes use of Q-values for each feasible action to take for the given State.
3) To achieve higher rewards, the agent chooses and carries out an action based on the activity's Q-Value.
4) Evaluate the reward and subsequent actions.
5) Save the earlier experience in the memory used for experience replay.
6) experience replay memory is used to train the network s.
7) For each State, repeat steps 2 through 6.

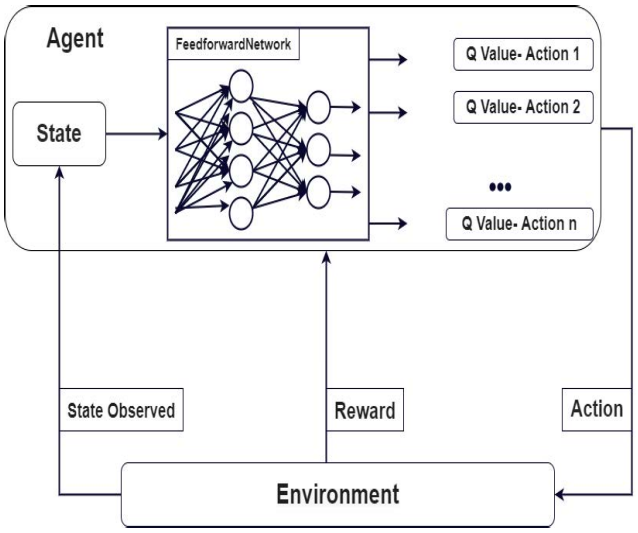


FIGURE 11. QL-Feed forward network block diagram.

Figure 10 shows the flowchart for the proposed method, and Figure 11 represents the block diagram of the proposed algorithm QFFR. The graphical abstracted view of the proposed algorithm QFFR is as shown in Figure 12

I. TIME COMPLEXITY

An acceptable Q-learning algorithm that uses an action-penalty representation finishes after at most O(n) steps after acquiring the goal State. A Markov Decision Process(MDP) comprises four parameters: S, A, P, and R [50] where,

S – is a set of States that an agent can be in at any particular time

A – is a set of conceivable actions that an agent can take at any particular time

P – is a set of possibilities that an agent in State' s' would transit to State 's' in 't+1' time by executing action A and

FIGURE 12. Generic view of QL-Feed forward routing algorithm.

R - Reward received by the agent as a result of taking some specific action and changing its State from s to s'.

When an action $a \epsilon A$ is taken in a State $s \epsilon S$, a reward R[s, a] is generated and the transitions to the next State $s \epsilon S$ with the probability of transitioning from s to s' Stated as given In equation 6.

$$P^a\_ss' = P_r S_{t-1} = S' | S_t = S, a_t = a \tag{6}$$

The total reward(cumulative) can be Stated as mentioned in equation 7.

$$R_t otal = \sigma(\sum_{t=o}^{\infty} \gamma^t r_t) \tag{7}$$

The proposed QLFF combines a Q-learning algorithm with a Feed Forward network. The complexity of exploring unexplored grid worlds is very low. State space is referred to as 1-step invertible if it has no duplicate actions, is Stated in equation 4 for any SϵS and $a \epsilon A(s)$, and has no duplicate actions. Given that the input neurons represent the inputs, the only neurons in feed-Forward networks that carry out operations are the hidden and output neurons (so they do not perform any process). Each hidden neuron first applies a non-linear (or activation) function to a linear combination of its inputs. As a result, every hidden neuron executes the operation as indicated in equation 8.

$$y_j = \sigma(\sum_{i}^{n} w_{ij} x_i) \tag{8}$$

where i represents the input from the input neuron i, w_{ij} represents the weight of the connection from the input neuron i to the hidden neuron j, and y_j is used to represent the output of neuron j. Thus, the worst-case complexity of attaining a goal State has a tight bound of $O(n^3)$ if a proper task representation or appropriate initialization is chosen. For the value-iteration action executions and the Q-learning action executions, respectively, the time complexity is $O(n^2)$. The $O(n^3)$ bound can be further reduced if the agent has prior knowledge of the topology of the State space or the State space has extra features [58], [59].

IV. EXPERIMENTAL SET UP AND RESULT ANALYSIS

A. EXPERIMENTAL SET UP FOR TRADITIONAL ALGORITHMS (SIMULATION SET UP)

The network simulator (ns3) version 3.25 is used for observing the performance of four traditional routing protocols, namely, Ad hoc On-Demand Distance Vector (AODV), Optimized Link State routing (OLSR), Destination-Sequenced Distance-Vector (DSDV) and Distance Source routing (DSR). They are compared using the network Simulator, which is said to be one of the standard simulation tools available [22]. ns3's simulation environment is set up on the Ubuntu 20.04 LTS operating system. The simulation lasts 200 seconds, the first 50 of which serve as start-up time, where a popular random waypoint mobility model is used. Node density is an essential factor that significantly impacts how well routing algorithms work. A simulation environment with 50 nodes is set to confirm the network's thoroughness. Within a 300 × 1500 m area, the nodes move with a speed of 20 m/s and no pause time following the RandomWaypoint-Mobility Model. With a 2 Mb/s rate (802.11b) and a Friis loss model, the Wi-Fi operates in ad hoc mode. The transmission power is 7.5 dBm. Ten source/sink data pairs transmit the data at a rate of 2.048 Kbps each. Usually, 64-byte packets are sent per second in this manner. The simulations can be conducted by directly changing the speed and quantity of nodes, it is possible to vary the network's density and mobility. Changing the transmit power allows users to alter the network's properties as well (as power increases, the impact of mobility decreases and the effective density increases), which can give produce variation in algorithm's performance [60]. Table 2 shows the parameters selected while setting up the experiment. Communication between the nodes is established at random. The source and destination nodes, and also the duration of the connections between nodes, are generated at random using the uniform random variable defined in ns3.

1) MOBILITY MODEL

The nodes transverse according to the Random Waypoint-Mobility Model in this experiment. In mobility management, the random WaypointMobility model is used. It is a random model for mobile user movement, including how their velocity, acceleration and location change over time. Random-based mobility simulation models have mobile nodes that move at random and without restriction. More accurately, the destination, speed, and direction are all chosen at random, with no consideration for the other nodes. The network's mobility and density can be varied by adjusting the network's speed and the number of nodes. The transmit power can also be changed to change the network's characteristics (as power

TABLE 2. Description of NS3 parameters for experiment set up.

Sr.No	Parameter	value
1	Operating System:	Ubuntu 20.04 LTS
2	network Simulator:	ns-3.30.10
3	Simulation Time:	200 sec simulation runs for 200 simulated seconds of which the first 50 are used for startup time.
4	Number of nodes:	50
5	No of sinks:	10
6	Mobility Model:	Random WaypointMobility Model
7	Propagation Model:	Constant Speed Propagation Delay
8	Propagation Loss Model:	Friis
9	Position Allocator:	Random Rectangular Position Allocator
10	Mac:	Ad-hoc Wi-Fi MAC
11	Mac Standard:	802.11B
12	Bps:	2 Kbps
13	node speed:	20m/s
14	node pause time:	0
15	Transmit power	7.5 dBm

increases, the impact of mobility decreases and the effective density increases). The Wi-Fi is in ad hoc mode, with a rate of 2 Mb/s (802.11b) and a Friis loss model. The Friis free space propagation model simulates Line of sight(LOS) route loss in a free space environment free of objects that cause the characteristic-altering phenomenon to a radiated wave like absorption, diffraction, and reflections.

2) PROPAGATION LOSS MODEL

The loss of power or attenuation of a signal or radio wave travelling via a transmission channel can be simulated using a propagation loss model. It facilitates the estimation of a destination node's reception power. This permits them to see if the node can receive a signal. The reception power is determined by the source node's emission power as well as the position of the source and destination nodes. The position of nodes is determined by the mobility modes established. In this case, the Long-distance Propagation Loss model is used. The reception power can be calculated as given in Equation.

$$PL = PL_0 + 10_n log10(d/d_0) \tag{9}$$

where,
d_0: reference distance (m)
PL_0: path loss at the reference distance (dB)
d: distance (m)
PL: path loss (dB)

B. RESULT-TRADITIONAL ALGORITHMS

Table 3 shows the results of the performance of four traditional routing protocols, namely, Ad hoc On-Demand Distance Vector (AODV) and Optimized Link State routing (OLSR) Destination-Sequenced Distance-Vector (DSDV) and Distance Source routing (DSR). As per the set of parameters mentioned in Table 2, the experiments are conducted for four algorithms, i.e. AODV, OLSR, DSDV and DSR. The analysis is accomplished using the parameters like packet rate(packets received per second), bits per second at every flow(packet flow), delay and the number of lost packets.Figures 13, 15, 17, and Figure 19 illustrate the AODV, OLSR, DSDV, and DSR packet receiving rates respectively. Each flow is a collection of packets consisting same protocol, source (IP, port), and destination (IP, port. Packets are categorized based on the flow to which they belong. Figures 14, 16, and 18, respectively, illustrate the number of packet flows for AODV, OLSR, and DSDV and the total end-to-end delays for all packet flows.The overall comparison based on packets received per second for the mentioned algorithms is seen in Figure 20. Throughput, latency, and packet delivery ratio (PDR) are the performance measures used in the evaluation.

TABLE 3. Result analysis of traditional algorithms.

Sr.No	Algorithm	Throughput (Kbps)	Delay (ms)	Packet Delivery Ratio
1	AODV	723.13	343.73	71.25
2	DSDV	259.11	496.76	67 25
3	OLSR	752.81	466.23	74.67
4	DSR	743.56	428.58	72.65

1) PACKET DELIVERY RATIO (PDR)

The proportion of packets transmitted by the application to packets received at the destination.

2) THROUGHPUT

The number of packets transmitted to a destination divided by the number of packets received is known as throughput. In a highly dynamic topology where the viability of a route can quickly change, this is an excellent indicator to assess a route's quality.

3) DELAY

The amount of time it takes a packet from its source node to reach the destination node. The packet's delay is estimated from when it leaves the source node until it reaches its destination. As stated earlier, in these algorithms, the router's decision is as per the protocol's policy. Due to this, they take the same conclusion, facing similar challenges at times. Table 3 shows the algorithms' performance as per the specifications mentioned in Table 2. Table 3 illustrates how AODV outperforms in the given experimental set-up. OLSR has a throughput of 723.13 Kbps, AODV of 752.81 Kbps, DSDV of 259.11 Kbps, and DSR of 743.56 Kbps. It arises due to AODV's reduced bandwidth usage compared to the

competition [19]. Table 3 contains the analysis for packet Delivery Ratio AODV, OLSR, DSDV, and DSR. The Packet Delivery Ratio of OLSR is 4.580 percent better than AODV, 10.272 percent better than DSDV, and 2.705 per cent better than DSR, according to the study's findings. OLSR can maintain performance despite an increase in the number of nodes [19]. It is evident since OLSR is proactive and regularly broadcasts messages. It always maintains the informational route for the entire network and is immediately usable for packet transmission references and a few more. Compared to DSDV, OLSR, and DSR, AODV has a better average delay. According to the Table3, the average delay for AODV is 343.73 ms, OLSR is 466.23 ms, DSDV is 496.76 ms, and DSR is 428.58 ms.

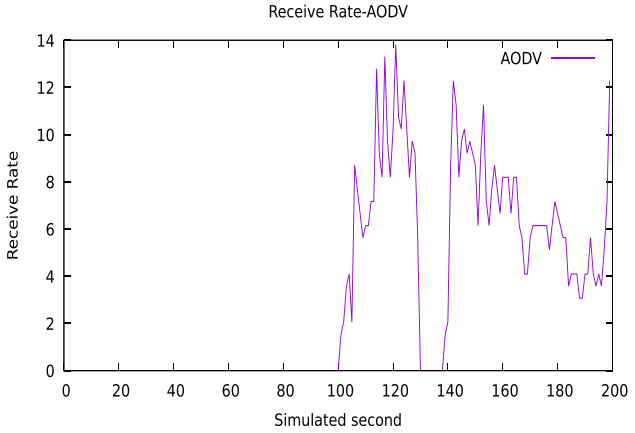


FIGURE 13. Performance of AODV Algorithms 1.

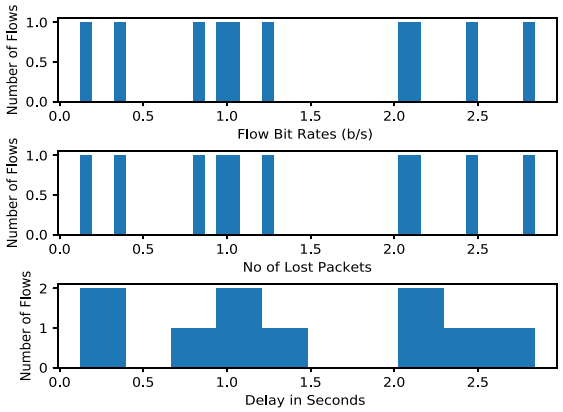


FIGURE 14. Performance of AODV Algorithms 2.

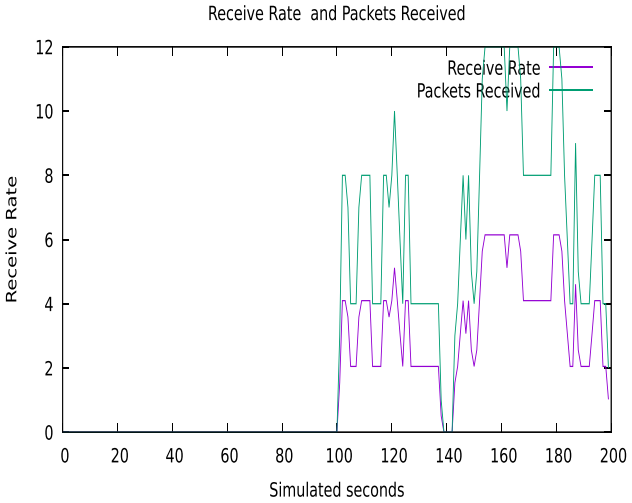


FIGURE 15. Performance of OLSR Algorithms 1.

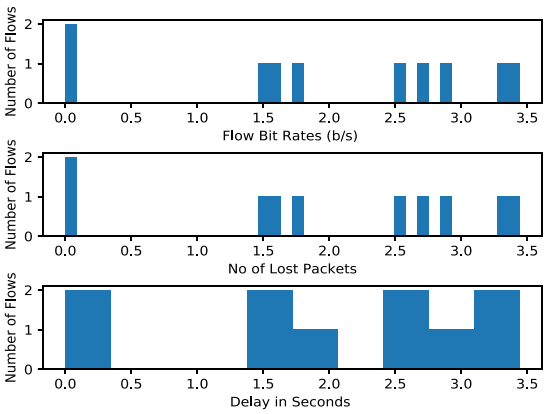


FIGURE 16. Performance of OLSR Algorithms 2.

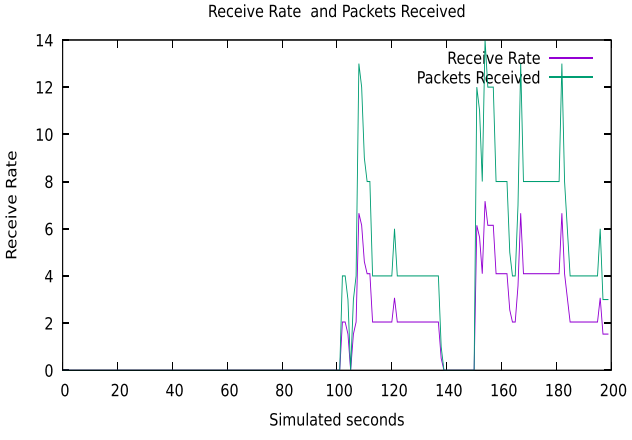


FIGURE 17. Performance of DSDV Algorithms 1.

C. THE ENVIRONMENT SETUP FOR Q-LEARNING AND PROPOSED QL-FEED FORWARD ROUTING ALGORITHM (QFFR)

The wireless mesh network is formed using a grid structure of three by four size, considered a sample network. Python has been used as a programming language to work with high-dimensional environments like those offered by the OpenAI Gym. The experiment uses a new open-source package, Simple rl, for conducting reinforcement-based (Q-learning) algorithm' experiments [61]. The essential three components in this experiment are routers, RL agents (or a collection of agents), and an environment (an MDP)

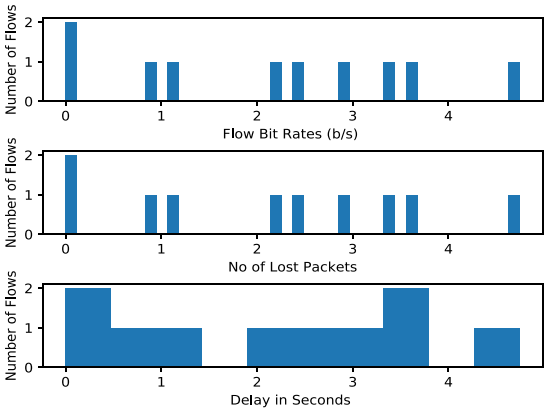


FIGURE 18. Performance of DSDV Algorithms 2.

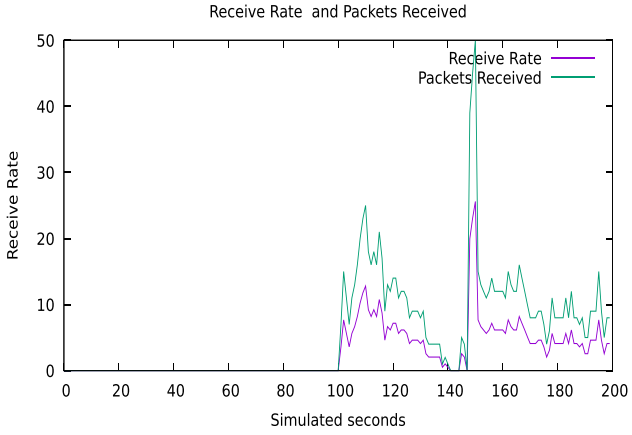


FIGURE 19. Performance of DSR Algorithms 1.

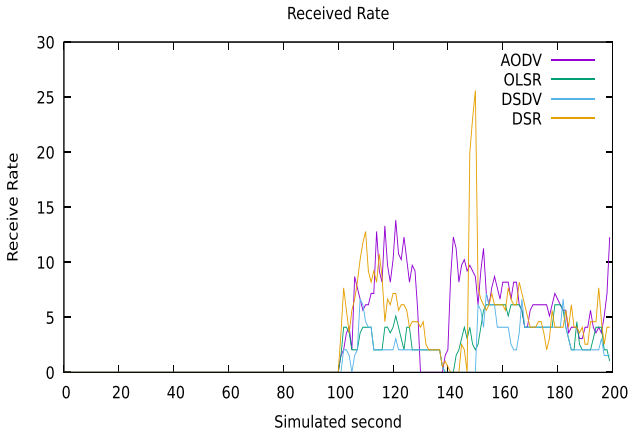


FIGURE 20. Comparison of traditional algorithms.

that models the wireless mesh environment. The agent (s) communicate with the environment to make the routing decision. The QL-Feed Forward routing algorithm (QFFR) experiment is conducted considering two scenarios. In the first scenario, the routers are arranged in a grid format, i.e. a grid architecture formed by three rows by four columns with each router at each intersection as shown in Figure 8. In the second scenario, the mesh network is formed using the python library network x instead of considering a grid architecture, as shown in Figure 25. The shortest path is computed, which shows the time taken by the agent to reach the destination. Identification of the shortest path (faster packet delivery) is a significant factor in the routing process from the network s' performance perspective. The time taken to reach the destination significantly impacts performance metrics like throughput, delay, packet receiving rate, and the packet delivery ratio.

A taxi-v3 gym environment is chosen for the proposed QL-Feed Forward routing algorithm (QFFR). This is a simple environment where a router acts as an agent and must pick up and drop off packers for transmission while accomplishing six different activities while sending the packets from source to destination. The experiment is carried out using various combinations of learning rate alpha (*alpha*) and discount factor gamma (*gamma*).

1) STATE SPACE

State Space S is a collection of all possible States for an agent. The nodes in this experiment are considered to form a grid-like layout regarded as a mesh architecture, and the scenario is handled using the well-known World Grid Problem [62]. As indicated in Figure 8, a grid with three rows and four columns is explored. As discrete States, this yields in a State-space S formed by $3 \times 4 = 12$. Node (router) R5 is the source node (second row, first column). Node (router) R8 is the target (destination) node (second row, fourth column).

2) ACTION SPACE

The collection of all conceivable actions 'a' that an agent can execute is known as the action space A [11]. This includes actions to Pick up the packet, deliver the packet, move up, move down, move left, and move right).

3) TRANSITION FUNCTION

When an agent performs an action a, the transition function T(S'| S, a) characterizes how an agent transitions from current State S to S'. For a given State S, this function provides the probability distribution across all possible following States S' (S, a) [11].

4) REWARD FUNCTION

The reward that an agent receives when performing action 'a' from State S, R is defined by a reward function (S, a). There can be a Good action yielding a reward or a Bad action yielding No reward or penalty [11].

5) DISCOUNT FACTOR

The discount factor gamma γ in the range (0,1), determines how myopic (short-sighted) the agent is when making decisions. (Discount factor, =0 short-sighted, and =1 long-sighted) A short-sighted decision is one in which the agent is mainly concerned with current benefits. In contrast, a long-sighted decision is one in which the agent includes potential future information [63], [64].

6) LEARNING RATE

The agent gathers information from its environment to forecast future actions. To increase the accuracy of these predictions, new information must be incorporated into the learning process to reflect the current State of the environment. The prediction error, which assesses the difference between the current forecast and the observed environmental State, and the learning rate alpha α, which defines how much of the prediction error is applied to updating the prediction, are two crucial aspects in the integration of new information. The learning rate alpha determines how much the model updates the action value depending on the reward prediction error [63], [64].

V. RESULTS

A. Q LEARNING WITH GRID ARCHITECTURE

The results of the Q learning algorithm applied to the grid network architecture, with various values for instances, episodes and steps to analyse the performance of the Q learning algorithm, are discussed as given below. In this experiment, the Instances are the number of times the experiment should be repeated. Episodes are the number of episodes per instance. An episode will consist of a sequence of steps, after which the agent will be reset to its initial condition (but gets to remember what it has learned so far). The Steps are the number of steps in each episode. The agent chooses the action to move to the next State (node) with the maximum reward. With the different number of episodes and number of steps analysed in the experiment. Figure 21 depicts Q learning on a three-by-four grid with a step size of 10. Figure 22 illustrate the results of Q-learning with three by four grid architecture, number of episodes set to 100 and step size set to 10. Figure 23 illustrate the results of Q-learning with three by four grid architecture, number of episodes set to 100 and step size set to 20. Figure 24 shows the Q learning output with three by four grid architecture where the experiment repeated with 50 episodes and step size 10. Changes to the grid size, the number of episodes, and the agent's steps to explore the environment, as shown in case1,case2 and case3 are used to validate the performance. The results obtained are stated as below:

1) Case 1: In this analysis, the number of episodes is fifty(50), and steps are defined as equal to ten(10). Figure 21 shows the cumulative rewards achieved by the Q-learning agent. The time taken by the agent is shown in Table 4. The agent has taken 0.49 sec to complete the Q-learning.
2) Case 2: In this analysis, the number of episodes is hundred (100), and steps are defined as equal to ten(10). Figure 22 shows the cumulative rewards achieved by the Q-learning agent. The time taken by the agent is shown in Table 4. The agent has taken 0.53 sec to complete the Q-learning. Figure 23 shows the performance of the QL agent with hundred episodes and a step size equal to 20.

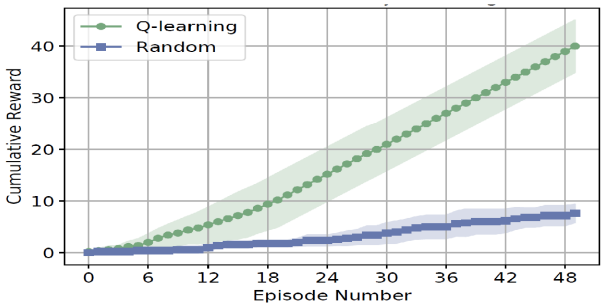


FIGURE 21. Q learning repeated experiment with ten by ten grid, episode = 50 step = 10.

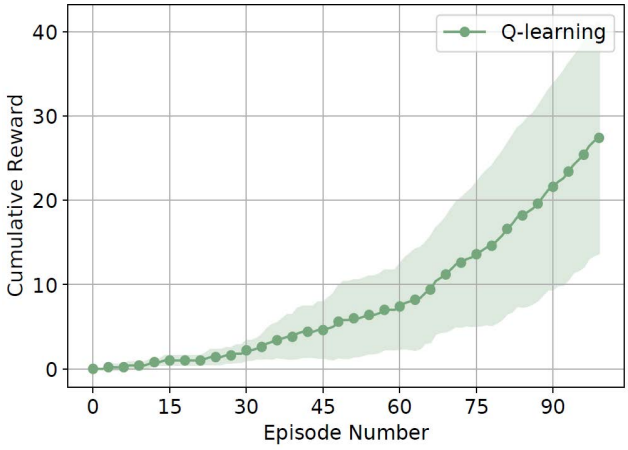


FIGURE 22. Q learning with three by four grid, episode = 100 step = 10.

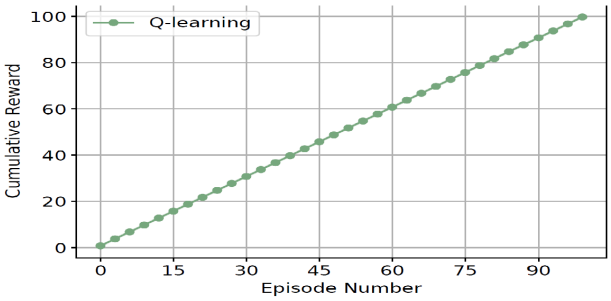


FIGURE 23. Q-learning with three by four grid, episode = 100 step = 20.

3) Case 3: Figure 24 shows the agent's performance when the experiment is repeated. The time taken by the agent is shown in Figure Table 4. The agent has taken 0.47 seconds to complete the Q leaning, which is nothing but time taken to choose a set of actions to accomplish the task.

B. Q-LEARNING WITHOUT GRID ARCHITECTURE

Many times, in WMNs, the nodes may not form the grid architecture. So, a random network forming a mesh of nodes is generated using the well-known python library "networkx", as shown in Figure 25, to test the performance of the Q learning algorithm. The time the algorithm takes when applied to

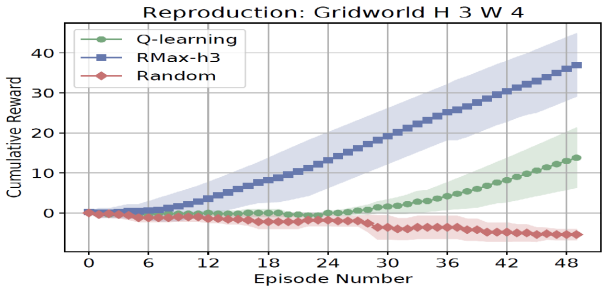


FIGURE 24. Q learning with repeated experiment with episode = 50, step = 10.

TABLE 4. Performance of Q-learning algorithm.

Learning Agent	Time (sec) Grid size(10X10) Ten by Ten	Time(sec) Grid size (3X4) Three by Four
Q-Learning Agent	0.47 S	0.53
R max Agent	3.3	3.5
Random Agent	0.32	0.41

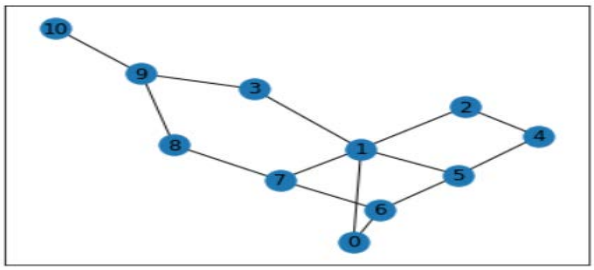


FIGURE 25. Wireless mesh network formed by nodes.

non-grid architecture is computed. The experiment was run by varying the discount factorgamma(*gamma*) values equal to 0.9,0.99(almost equal to 1) and 0.6 (below 1).Table 5 shows the graphical output and time taken by the agent to complete the task which is 3.5 sec, 4.0 sec and 3.7 sec for gamma(*gamma*) values equal to 0.9,0.99(almost equal to 1) and 0.6respectively.

C. RESULT PROPOSED QL-FEED FORWARD ROUTING ALGORITHM (QFFR)

The results of proposed algorithm are shown in the Table 7. The parameters Such as Total episodes, Total test episodes, Max steps per episode, Exploration rate, Exploration probability at the start, Minimum exploration probability and Exponential decay rate for exploration probability are set as described in Table 7 for the conduction of this experiment. The suggested algorithm provides an environment in which the agent (router) can communicate using a trial-and-error method with 5000 episodes, of which there are 100 test episodes. The agent's exploration rate epsilon (ϵ), which defines how frequently it should explore and exploit, is set to 1. Too little or too much exploration may be ineffective

TABLE 5. Results for mesh network formed without grid architecture.

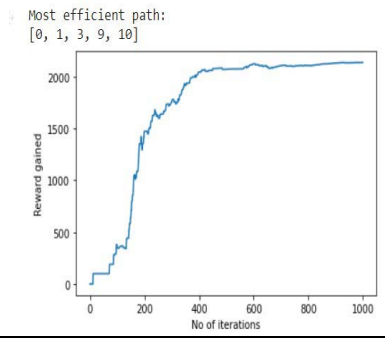
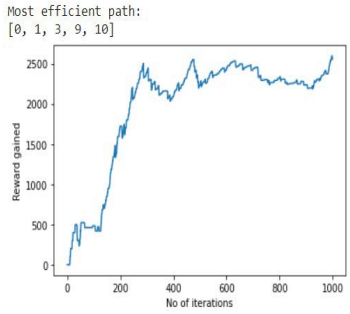
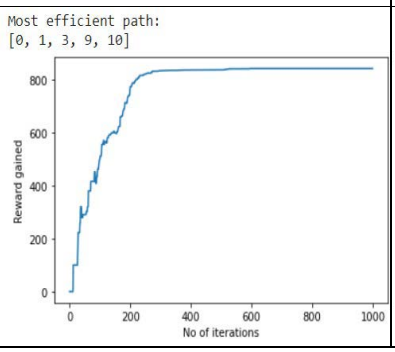
Parameter Set	Graphical Output	Time taken
Gamma=0.9		3.5sec
Gamma=0.99		4 sec
Gamma=0.6		3.7 sec

TABLE 6. Parameters set for proposed algorithm.

Description	Parameter	value
Total episodes	total-episodes	5000
Total test episodes	total-test-episodes	100
Max steps per episode	max-steps	99
Exploration rate	epsilon	1
Exploration probability at start	max-epsilon	1
Minimum exploration probability	min -epsilon	0.01
Exponential decay rate for exploration prob	decay-rate	0.01

for the agent to learn from the environment. Hence Start maxepsilon's exploration probability is set at 1 and 0.01 minimum. The exponential decay rate for exploration probability is adjusted to 0.01 to stabilize the model's output. The results of the suggested algorithm Q-learning feed-forward (QLFF) are shown in table 7. The various parameters and

TABLE 7. Results of proposed algorithm.

Case Number	Discount factor (gamma)	Learning Rate (alpha)	Score over time
1	0.5	0.7	7.86
2	0.8	0.7	8.07
3	0.618	0.7	7.93
4	0.618	0.8	7.62
5	0.618	0.5	7.62

their values are shown in Table 6. The time taken by the agent to complete the task is shown below, as mentioned in five different cases. The time taken by the algorithm to complete the task with variation in the discount factor gamma *gamma* and the learning rate alpha *alpha* are as mentioned below:

1) Case 1: Discount factor = gamma = 0.5, Learning Rate (alpha) = 0.7 Here the parameters are set up as shown in Table 6. The time taken to complete the task by the proposed model is 7.86 ns, as shown in Table 7.
2) Case 2: Discount factor = gamma = 0.8, Learning Rate (alpha) = 0.7 Here the parameters are set up as shown in Table.6 The time taken to complete the task by the proposed model is 8.07 s s shown in Table 7.
3) Case 3: Discount factor = gamma = 0.618, Learning Rate (alpha) = 0.7 Here the parameters are set up as shown in Table.6 The time taken to complete the task by the proposed model is 8.1 s s shown in Table 7.
4) Case 4: Discount factor = gamma = 0.618, Learning Rate (alpha) = 0.8 Here the parameters are set up as shown in Table.6 The time taken to complete the task by the proposed model is 7.62 s s shown in Table 7
5) Case 5: Discount factor = gamma = 0.618, Learning Rate (alpha) = 0.5 Here the parameters are set up as shown in Table 6. The time taken to complete the task by the proposed model is 7.62 s s shown in Table 7

The combinations of discount factor gamma *gamma* and the learning rate alpha *alpha* are used to understand their impact on the time taken for task completion.

VI. CONCLUSION

This paper presents a detailed analysis of traditional and latest Reinforcement Learning-based routing algorithms' implementation to select the best route in wireless mesh networks. Two reactive namely, Dynamic Source routing (DSR) and Destination Sequenced Distance Vector (DSDV) algorithms and two proactive, namely, Ad hoc On-Demand Distance Vector(AODV) and Optimized Link State routing (OLSR) algorithms, have been analyzed among the traditional routing algorithms. This paper also shows the implementation of the Q learning algorithm and suggests a hybrid QL-Feed Forward routing algorithm (QFFR) algorithm for routing in wireless mesh networks.AODV and DSR show good performance regarding packets received and packet loss ratio among the traditional algorithms. OLSR has a throughput of 723.13 Kbps, AODV of 752.81 Kbps, DSDV of 259.11 Kbps, and DSR of 743.56 Kbps. The Packet Delivery Ratio of AODV is 71.25, OLSR is 74.67, DSDV is 67.25, and DSR is 72365. The average delay for AODV is 343.73 ms, OLSR is 466.23 ms, DSDV is 496.76 ms, and DSR is 428.58 ms.Implementation of Q learning algorithm shows that the Q learning agent completes the task of identification of route and reaches to the destination in average of in 3.7s in non-grid architecture. The Q-learning agent takes 0.49 sec with a grid size ten by ten and 0.53sec in three by four grid size. The suggested QLFFR takes average of 0.7 sec score over time as observed. The fundamental difference between the traditional and Reinforcement Learning-based algorithms is that the Reinforcement Learning algorithms decide to select the route(in turn, next-hop) as per their learnings from the environment. The learning rate and discount factors significantly impact the time the algorithm selects the entire route. The Q learning for routing gives the path identification for the given network faster than the QL-Feed Forward routing algorithm (QFFR). The suggested QL-Feed Forward routing algorithm (QFFR) has an advantage over the Q learning implemented for routing because the increase in the number of nodes spanning the size of the network architecture will not affect the performance. Combining Q learning algorithm with Long Short Term Memory or Convolution Neural networks to enhance the wireless network's performance could be the future scope of this work. Further, the design of the customized network architecture or test-bed formed for the wireless mesh network, rather than using the predefined architectures, is the need of the present time. An interesting point to note is that ns-3 is more modular than specific simulation platforms, which offer users a single, integrated graphical user interface environment in which all operations related to networking are accomplished. However, command-line work and the use of Python or C++ software development tools are anticipated by users. Most AI algorithms presumably use open-source frameworks like TensorFlow and PyTorch. There are a few other machine learning-specific libraries that ns3 does not support. However, the most recent version of ns3-gym is a framework that combines OpenAI Gym with ns-3 to foster the use of RL in networking research, which may be the future application of this work.

REFERENCES

[1] K. Andreev and P. Boyko, *IEEE 802.11S Mesh Networking NS-3 Model*.
[2] S. Banerji and R. S. Chowdhury, "On IEEE 802.11: Wireless LAN technology," *Int. J. Mobile Netw. Commun. Telematics*, vol. 3, pp. 45–64, Aug. 2013.
[3] K. Leevangtou, H. Ochiai, and C. Aswakul, "Application of Q-learning in routing of software-defined wireless mesh network," *IEEJ Trans. Electr. Electron. Eng.*, vol. 17, no. 3, pp. 387–397, Mar. 2022.
[4] Y. Chai and X.-J. Zeng, "Regional condition-aware hybrid routing protocol for hybrid wireless mesh network," *Comput. Netw.*, vol. 148, pp. 120–128, Jan. 2019.
[5] L. L. Maria, *Whitireia NZ: Faculty of Business and Information Technology Security Challenges in Wireless Mesh Networks-Literature Review Academic Essay Security Challenges in Wireless Mesh Networks-Literature Review*.

[6] Q. Mao, F. Hu, and Q. Hao, "Deep learning for intelligent wireless networks: A comprehensive survey," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 4, pp. 2595–2621, 4th Quart., 2018.

[7] R. Boutaba, M. A. Salahuddin, N. Limam, S. Ayoubi, N. Shahriar, F. Estrada-Solano, and O. M. Caicedo, "A comprehensive survey on machine learning for networking: Evolution, applications and research opportunities," *J. Internet Services Appl.*, vol. 9, no. 1, Dec. 2018.

[8] B. Mao, Z. Md Fadlullah, F. Tang, N. Kato, O. Akashi, T. Inoue, and K. Mizutani, "Routing or computing? The paradigm shift towards intelligent computer network packet transmission based on deep learning," *IEEE Trans. Comput.*, vol. 66, no. 11, pp. 1946–1960, Nov. 2017.

[9] B. Sahu, P. K. Das, M. R. Kabat, and R. Kumar, "Prevention of COVID-19 affected patient using multi robot cooperation and Q-learning approach: A solution," *Quality Quantity*, vol. 56, no. 2, pp. 793–821, Apr. 2022.

[10] G. George, R. Karim Lakhani, and P. Puranam, "What has changed? The impact of COVID pandemic on the technology and innovation management research agenda," *J. Manag. Stud.*, vol. 57, no. 8, pp. 1–6, Dec. 2020.

[11] D. R. Militani, H. P. D. Moraes, R. L. Rosa, L. Wuttisittikulkij, M. A. Ramírez, and D. Z. Rodríguez, "Enhanced routing algorithm based on reinforcement machine learning—A case of VoIP service," *Sensors*, vol. 21, pp. 1–32, Jan. 2021.

[12] Y. Chai and X.-J. Zeng, "The development of green wireless mesh network: A survey," *J. Smart Environ. Green Comput.*, vol. 1, no. 1, pp. 47–59, Mar. 2021.

[13] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, G. M. Bellemare, A. Graves, M. Riedmiller, K. A. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, pp. 529–533, Feb. 2015.

[14] R. Ding, Y. Xu, F. Gao, X. Shen, and W. Wu, "Deep reinforcement learning for router selection in network with heavy traffic," *IEEE Access*, vol. 7, pp. 37109–37120, 2019.

[15] S. P. M. Choi and D.-Y. Yeung, "Predictive Q-routing: A memory-based reinforcement learning approach to adaptive traffic control," in *Proc. Adv. Neural Inf. Process. Syst.*, 1995, pp. 1–7.

[16] Y. Sun, M. Peng, Y. Zhou, Y. Huang, and S. Mao, "Application of machine learning in wireless networks: Key techniques and open issues," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 4, pp. 3072–3108, 4th Quart., 2019.

[17] S. Fujimoto, H. Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 1587–1596.

[18] A. Cilfone, L. Davoli, L. Belli, and G. Ferrari, "Wireless mesh networking: An IoT-oriented perspective survey on relevant technologies," *Future Internet*, vol. 11, no. 4, p. 99, Apr. 2019.

[19] Alamsyah, M. H. Purnomo, I. K. E. Purnama, and E. Setijadi, "Performance of the routing protocols AODV, DSDV and OLSR in health monitoring using NS3," in *Proc. Int. Seminar Intell. Technol. Appl. (ISITIA)*, Jul. 2016, pp. 323–328.

[20] H. M. Haglan, S. A. Mostafa, N. Z. M. Safar, A. Mustapha, M. Z. Saringatb, H. Alhakami, and W. Alhakami, "Analyzing the impact of the number of nodes on the performance of the routing protocols in MANET environment," *Bull. Electr. Eng. Informat.*, vol. 10, no. 1, pp. 434–440, Feb. 2021.

[21] Q. Liu, L. Cheng, A. L. Jia, and C. Liu, "Deep reinforcement learning for communication flow control in wireless mesh networks," *IEEE Netw.*, vol. 35, no. 2, pp. 112–119, Mar. 2021.

[22] S. D. Samo and J. L. E. K. Fendji, "Evaluation of energy consumption of proactive, reactive, and hybrid routing protocols in wireless mesh networks using 802.11 standards," *J. Comput. Commun.*, vol. 6, no. 4, pp. 1–30, 2018.

[23] M. M. A. Alkadhmi, O. N. Uçan, and M. Ilyas, "An efficient and reliable routing method for hybrid mobile Ad hoc networks using deep reinforcement learning," *Appl. Bionics Biomechanics*, vol. 2020, pp. 1–13, Dec. 2020.

[24] Q. Fu, E. Sun, K. Meng, M. Li, and Y. Zhang, "Deep Q-learning for routing schemes in SDN-based data center networks," *IEEE Access*, vol. 8, pp. 103491–103499, 2020.

[25] H. Alavizadeh, J. Jang-Jaccard, and H. Alavizadeh, "Deep Q-learning based reinforcement learning approach for network intrusion detection," *Computers*, vol. 11, no. 3, p. 41, Mar. 2021.

[26] H. Bae, G. Kim, J. Kim, D. Qian, and S. Lee, "Multi-robot path planning method using reinforcement learning," *Appl. Sci.*, vol. 9, no. 15, p. 3057, Jul. 2019.

[27] T.-V.-T. Duong, L. H. Binh, and V. M. Ngo, "Reinforcement learning for QoS-guaranteed intelligent routing in wireless mesh networks with heavy traffic load," *ICT Exp.*, vol. 8, no. 1, pp. 18–24, Mar. 2022.

[28] S. Kaviani, B. Ryu, E. Ahmed, K. A. Larson, A. Le, A. Yahja, and J. H. Kim, "Robust and scalable routing with multi-agent deep reinforcement learning for MANETs," 2021, *arXiv:2101.03273*.

[29] X. You, X. Li, Y. Xu, H. Feng, J. Zhao, and H. Yan, "Toward packet routing with fully distributed multiagent deep reinforcement learning," *IEEE Trans. Syst., Man, Cybern. Syst.*, vol. 52, no. 2, pp. 855–868, Feb. 2022.

[30] Z. A. Khan, O. A. Karim, S. Abbas, N. Javaid, Y. B. Zikria, and U. Tariq, "Q-learning based energy-efficient and void avoidance routing protocol for underwater acoustic sensor networks," *Comput. Netw.*, vol. 197, Oct. 2021, Art. no. 108309.

[31] V. Di Valerio, F. L. Presti, C. Petrioli, L. Picari, D. Spaccini, and S. Basagni, "CARMA: Channel-aware reinforcement learning-based multi-path adaptive routing for underwater wireless sensor networks," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 11, pp. 2634–2647, Nov. 2019.

[32] T. Saba, K. Haseeb, A. A. Shah, A. Rehman, U. Tariq, and Z. Mehmood, "A machine-learning-based approach for autonomous IoT security," *IT Prof.*, vol. 23, no. 3, pp. 69–75, May 2021.

[33] M. Abdollahi, W. Ni, M. Abolhasan, and S. Li, "Software-defined networking-based adaptive routing for multi-hop multi-frequency wireless mesh," *IEEE Trans. Veh. Technol.*, vol. 70, no. 12, pp. 13073–13086, Dec. 2021.

[34] K. Haseeb, I. Ud Din, A. Almogren, N. Islam, and A. Altameem, "RTS: A robust and trusted scheme for IoT-based mobile wireless mesh networks," *IEEE Access*, vol. 8, pp. 68379–68390, 2020.

[35] C. Schüler, M. Patchou, B. Sliwa, and C. Wietfeld, "Robust machine learning-enabled routing for highly mobile vehicular networks with PARRoT in ns-3," in *Proc. Workshop ns-3*, Jun. 2021, pp. 88–94.

[36] B. Sharma and A. Singh, "Routing protocol for wireless mesh network—A survey," *Adv. Appl. Math. Sci.*, vol. 18, no. 8, pp. 1–12, 2019.

[37] R. Malekian, A. Karadimce, and A. H. Abdullah, "AODV and OLSR routing protocols in MANET," in *Proc. IEEE 33rd Int. Conf. Distrib. Comput. Syst. Workshops*, Jul. 2013, pp. 286–289.

[38] V. H. Nguyen, V. H. Nam, D. M. Linh, and V. K. Quy, "An improved agent-based AODV routing protocol for MANET," *EAI Endorsed Trans. Ind. Netw. Intell. Syst.*, vol. 8, pp. 1–8, May 2021.

[39] A. M. El-Semary and H. Diab, "BP-AODV: Blackhole protected AODV routing protocol for MANETs based on chaotic map," *IEEE Access*, vol. 7, pp. 95197–95211, 2019.

[40] M. Duraipandian, "Performance evaluation of routing algorithm for MANET based on the machine learning techniques," *J. Trends Comput. Sci. Smart Technol.*, vol. 1, no. 1, pp. 25–38, Sep. 2019.

[41] B. Sliwa, S. Falten, and C. Wietfeld, "Performance evaluation and optimization of batman V routing for aerial and ground-based mobile Ad-hoc networks," in *Proc. IEEE 89th Veh. Technol. Conf. (VTC-Spring)*, Apr. 2019, pp. 1–7.

[42] M. Naravani, D. G. Narayan, S. Shinde, and M. M. Mulla, "A cross-layer routing metric with link prediction in wireless mesh networks," *Proc. Comput. Sci.*, vol. 171, pp. 2215–2224, 2020.

[43] E. Setijadi, I. K. E. Purnama, and M. H. Pumomo, "Performance comparative of AODV, AOMDV and DSDV routing protocols in MANET using NS2," in *Proc. Int. Seminar Appl. Technol. Inf. Commun.*, 2018, pp. 286–289.

[44] N. Kumari, S. K. Gupta, R. Choudhary, and S. L. Agrwal, "New performance analysis of AODV, DSDV and OLSR routing protocol for MANET," in *Proc. 3rd Int. Conf. Comput. Sustain. Global Develop. (INDIACom)*, 2016, pp. 33–35.

[45] R. Ahuja, A. B. Ahuja, and P. Ahuja, "Performance evaluation and comparison of AODV and DSR routing protocols in MANETs under wormhole attack," in *Proc. IEEE 2nd Int. Conf. Image Inf. Process. (ICIIP)*, Dec. 2013, pp. 699–702.

[46] T. H. Sureshbhai, M. Mahajan, and M. K. Rai, "An investigational analysis of DSDV, AODV and DSR routing protocols in mobile Ad hoc networks," in *Proc. Int. Conf. Intell. Circuits Syst. (ICICS)*, Apr. 2018, pp. 281–285.

[47] E. Gaona-García, S. Palechor-Mopán, L. Murcia-Sierra, and P. Gaona-García, "Implementation of the AODV routing protocol for message notification in a wireless sensor microgrid," in *Proc. Workshop Eng. Appl.* Springer, 2018, pp. 357–369.

[48] H. Al-Rawi, M. Ng, and K. Yau, "Application of reinforcement learning to routing in distributed wireless networks: A review," *Artif. Intell. Rev.*, vol. 43, no. 3, pp. 381–416, 2015.

[49] Z. Mammeri, "Reinforcement learning based routing in networks: Review and classification of approaches," *IEEE Access*, vol. 7, pp. 55916–55950, 2019.
[50] M. Abu Alsheikh, D. T. Hoang, D. Niyato, H.-P. Tan, and S. Lin, "Markov decision processes with applications in wireless sensor networks: A survey," *IEEE Commun. Surveys Tuts.*, vol. 17, no. 3, pp. 1239–1267, Apr. 2015.
[51] X. Wang and X. Wang, "Reinforcement learning-based multihop relaying: A decentralized Q-learning approach," *Entropy*, vol. 23, no. 10, p. 1310, Oct. 2021.
[52] J. Zhou, X. Gong, L. Sun, Y. Xie, and X. Yan, "Adaptive routing strategy based on improved double Q-learning for satellite Internet of Things," *Secur. Commun. Netw.*, vol. 2021, pp. 1–11, Apr. 2021.
[53] R. Bhattacharyya, A. Bura, D. Rengarajan, M. Rumuly, S. Shakkottai, D. Kalathil, R. K. P. Mok, and A. Dhamdhere, "QFlow: A reinforcement learning approach to high QoE video streaming over wireless networks," in *Proc. 20th ACM Int. Symp. Mobile Ad Hoc Netw. Comput.*, Jul. 2019, pp. 251–260.
[54] G. H. Erharter, T. F. Hansen, Z. Liu, and T. Marcher, "Reinforcement learning based process optimization and strategy development in conventional tunneling," *Autom. Construction*, vol. 127, Jul. 2021, Art. no. 103701.
[55] G. Lingam, R. R. Rout, and D. V. L. N. Somayajulu, "Adaptive deep Q-learning model for detecting social bots and influential users in online social networks," *Int. J. Speech Technol.*, vol. 49, no. 11, pp. 3947–3964, Nov. 2019.
[56] A. Iqbal and S. Aftab, "A feed-forward and pattern recognition ANN model for network intrusion detection," *Int. J. Comput. Netw. Inf. Secur.*, vol. 11, no. 4, pp. 19–25, Apr. 2019.
[57] H. J. Yalamanchi and B. Bose, "Reinforcement learning for network routing," Tech. Rep., 2007.
[58] S. Koenig and G. R. Simmons, "Complexity analysis of real-time reinforcement learning," in *Proc. 11th Nat. Conf. Artif. Intell.*, 1993, pp. 99–105.
[59] J. Fan, Z. Wang, Y. Xie, and Z. Yang, "A theoretical analysis of deep Q-learning," in *Proc. 2nd Conf. Learn. Dyn. Control*, 2019, pp. 486–489.
[60] X. Rui, "Performance analysis of mobile Ad hoc network routing protocols using NS-3 simulations," Ph.D. thesis, Univ. Kansas, Lawrence, KS, USA, 2019.
[61] D. Abel, "Simple_rl: Reproducible reinforcement learning in Python," in *Proc. ICLR*, 2019, pp. 1–11.
[62] X. Lu and H. M. Schwartz, "An investigation of guarding a territory problem in a grid world," in *Proc. Amer. Control Conf.*, Jun. 2010, pp. 3204–3210.
[63] J. Wu, J. Li, Y. Xiao, and J. Liu, "Towards cognitive routing based on deep reinforcement learning," Mar. 2020, *arXiv:2003.12439*.
[64] K. Katahira, "The relation between reinforcement learning parameters and the influence of reinforcement history on choice behavior," *J. Math. Psychol.*, vol. 66, pp. 59–69, Jun. 2015.

SMITA MAHAJAN (Member, IEEE) received the bachelor's degree from the Shivaji University of Maharashtra, and the master's degree in information technology from Mumbai University. She is currently pursuing the Ph.D. degree in computer science engineering with Symbiosis International University, Pune, Maharashtra, India. She is currently working as an Assistant Professor with the Computer Science Engineering Department, Symbiosis Institute of Technology, Symbiosis International Deemed University. Her main research interests include computer networking, wireless and broadband networks, deep learning, and machine learning.

R. HARIKRISHNAN (Senior Member, IEEE) received the bachelor's degree in electrical and electronics engineering from the University of Madras, the master's degree in energy system engineering from VIT University, Vellore, the master's degree in embedded system technologies from Anna University, Chennai, India, and the Ph.D. degree in electrical engineering from Sathyabama University, Chennai. He has 21 years of teaching, research, and industrial experience. He is currently working as an Associate Professor in Electronics and Telecommunication Engineering Department, Symbiosis Institute of Technology, Symbiosis International Deemed University, Pune, India. His main research interests include smart grid, the Internet of Things, artificial intelligence, and wireless sensor networks.

KETAN KOTECHA received the M.Tech. and Ph.D. degrees from IIT Bombay. He is currently the Head of the Symbiosis Centre for Applied AI (SCAAI). He has expertise and experience in cutting-edge research and projects in AI and deep learning for more than 25 years. He has published more than 100 articles widely in a number of excellent peer-reviewed journals on various topics ranging from cutting-edge AI, education policies, teaching-learning practices, and AI for all. He was a recipient of the two SPARC Projects worth INR 166 lacs from MHRD Government of India in AI in collaboration with Arizona State University, USA, and the University of Queensland, Australia, and also the recipient of numerous prestigious awards like the Erasmus+ Faculty Mobility Grant to Poland, the DUO-India Professors Fellowship for research in Responsible AI in collaboration with Brunel University, U.K., the LEAP Grant at Cambridge University U.K., the UKIERI Grant with Aston University U.K., and the Grant from Royal Academy of Engineering, the U.K. under Newton Bhabha Fund. He has published three patents and delivered keynote speeches at various national and international forums, including at the Machine Intelligence Laboratory, USA, at IIT Bombay under the World Bank Project, at the International Indian Science Festival organized by the Department of Science Technology, Government of India, and many more. Currently, he is an Associate Editor of IEEE Access journal.