Protocoles de routage basés sur l'apprentissage par renforcement pour l'optimisation de la durée de vie et de l'énergie des réseaux de capteurs sans fil

Elvis Obi

THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par l'Université Toulouse 3 - Paul Sabatier

Présentée et soutenue par

Elvis OBI

Le 10 juillet 2023

Protocoles de routage basés sur l'apprentissage par renforcement pour l'optimisation de la durée de vie et de l'énergie des réseaux de capteurs sans fil

---

Ecole doctorale : **EDMITT - Ecole Doctorale Mathématiques, Informatique et Télécommunications de Toulouse**

Spécialité : **Informatique et Télécommunications**

Unité de recherche :
IRIT : Institut de Recherche en Informatique de Toulouse

Thèse dirigée par
Zoubir MAMMERI et Jacques JORDA

Jury

M. Pascal LORENZ, Rapporteur
Mme Samia BOUZEFRANE, Rapporteure
M. Abderrezak RACHEDI, Examinateur
M. Aliyu DANJUMA USMAN, Examinateur
M. Zoubir MAMMERI, Directeur de thèse
M. Jacques JORDA, Co-directeur de thèse
M. Thierry GAYRAUD, Président

ACKNOWLEDGMENTS

I am grateful to God Almighty for His mercies, faithfulness, divine protection, and love throughout my life and the time of my research.

I owe a debt of gratitude to my honorary thesis director, Prof. Zoubir Mammeri, of the Paul Sabatier University and the Computer Science Research Institute of Toulouse (IRIT), Toulouse, France, for his unwavering support, wise counsel, and keen interest in both my research work and the creation of this manuscript. He helped me through the entire program with patience and perseverance.

I am particularly grateful to Dr. Jacques Jorda, Associate Professor, IRIT, Paul Sabatier University, Toulouse, who is my thesis co-supervisor for his encouragement and assistance. My sincere gratitude goes out to Dr. E. O. Ochia of the Department of Electrical and Computer Engineering at the University of Calgary in Calgary, Canada, for his enormous contribution to the thesis's software development and implementation foundation.

I am thankful to the reviewers and examiners who reviewed and/or examined my thesis and gave valuable comments.

I am very grateful to the Nigerian Petroleum Technology Trust Fund (PTDF) Overseas Scholarship Scheme for funding this research with the administration of Campus France. I am incredibly grateful to Ahmadu Bello University's administration in Zaria, Nigeria, for allowing me to take a study leave to begin this completed Ph.D. program. Many thanks to the Department of Electronics and Telecommunication Engineering, Faculty of Engineering, of Ahmadu Bello University, Zaria, for the opportunity and understanding to embark and finish this Ph.D. Thank you so much Prof. A.D. Usman for the support from the onset of this program and for taking out time to participate as one of the Examiners. I also say thank you to the Departmental HOD-Associate Professor A.M.ST. Tekanyi for your support.

Words cannot adequately explain the parental direction, encouragement, and support I had from the moment I was conceived, as well as the good actions and value my father, the late D.S. P. Paul Obi, and sweet mother, Mrs. Sussana Obi, deposited in my life.

I'm grateful to my gorgeous wife Mrs. Ogechi Patience Obi for her moral, material, and other assistance. May you prosper from your efforts and enjoy excellent health (Amen). I express heartfelt gratitude to my beloved and precious sons: Kingchurchill Edric Obi and Prince David Obi for their love and understanding. You will always be loved.

I also want to express my sincere gratitude to my siblings for their support. Thank you to my entire family for your generosity. May the All-Powerful God richly reward you.

To all my IRIT friends and colleagues: Engr. (Miss) Chaima Zoghlami, Dr. Damien Wojtowicz, Dr. Luis Lugo, Dr. Nabil El Malki, Dr. Malik Irain, Dr. Nathalie Neptune, Mr. Nischal Prasad and all those whose names are not mentioned, I thank you all for your support and encouragement.

Finally, Many thanks also go to my PTDF scholar friends: Engr. Timothy Oluwadero, Engr. Abdulrahman Olaniyan, Mr. Kenule Nwigbo Dr. Ajoku Chinedu, Dr. Usman Garba, and all those whose names are not mentioned for their support and encouragement.

Elvis OBI

RÉSUMÉ

L'utilisation écoénergétique des nœuds de capteurs est un défi majeur dans la conception des réseaux de capteurs sans fil (RCS). En effet, la durée de vie du réseau est déterminée par les sources d'énergie limitées des nœuds de capteurs dont le remplacement ou la recharge est presque impossible en raison de leur déploiement dans des environnements difficiles. Une façon efficace de prolonger la durée de vie du réseau est de concevoir un protocole de routage écoénergétique pour les RCS en utilisant l'intelligence artificielle telle que l'apprentissage par renforcement (RL) qui peut apprendre la dynamique du réseau des RCS. D'après la littérature, la majorité des protocoles de routage écoénergétique basés sur RL pour les RCS sont de nature distribuée. Bien que le protocole de routage basé sur RL distribué permette aux capteurs sans fil de s'adapter de manière adaptative à la nature dynamique changeante de l'environnement des RCS, ce qui entraîne une réduction de la complexité de calcul et du temps d'apprentissage du processus, il est toutefois limité à la recherche des chemins de routage optimaux globaux. Cela entraîne une dégradation de la durée de vie du réseau et de la consommation d'énergie. Un protocole de routage écoénergétique basé sur RL centralisé peut atténuer le défi de la recherche des chemins de routage optimaux globaux en raison de la vue globale des RCS. Cette thèse présente trois contributions qui sont présentées dans la suite.

Tout d'abord, cette thèse présente la conception d'un protocole de routage Lifetime-Aware Centralized Q-Routing Protocol (LACQRP) pour les RCS afin de maximiser la durée de vie du réseau. Ceci est réalisé en mettant en œuvre l'algorithme de Q-learning sur la passerelle des RCS, qui agit également en tant que contrôleur ayant une connaissance globale de la topologie du réseau. La passerelle génère tous les arbres couvrants de poids minimum possibles basés sur la distance (All-MSTs) qui forment l'ensemble des tables de routage (TR) pour un RCS composé de 100 nœuds de capteurs. La maximisation de la durée de vie du réseau est obtenue en apprenant au contrôleur les TR qui minimisent le maximum d'énergie de consommation des nœuds de capteurs. Les résultats de simulation montrent que le LACQRP apprend les TR optimales qui maximisent la durée de vie du réseau et présente un meilleur taux de convergence, une durée de vie de réseau et une consommation d'énergie moyenne lorsqu'il est comparé à certains protocoles de routage distribués basés sur la RL pour l'optimisation de la durée de vie (c'est-à-dire RL-Based Routing (RLBR) et RL for Lifetime Optimization (R2LTO)). Bien que LACQRP prolonge la durée de vie du réseau, le temps de calcul augmente de manière exponentielle avec le nombre de nœuds de capteurs. Par conséquent, cette méthode n'est pas pratique pour les RCS à grande échelle.

Deuxièmement, cette thèse propose également une méthode appelée Protocole de routage centralisé pour l'optimisation de la durée de vie avec l'algorithme génétique (GA) et Q-learning (CRPLOGAQL) pour réduire le temps de calcul de LACQRP. Pour ce faire, l'algorithme de All-MSTs de LACQRP est remplacé par les arbres couvrants de poids minimum proposés basés sur GA. Les simulations d'évaluation montrent que CRPLOGAQL peut fournir un routage sous-optimal avec un temps de calcul réduit par rapport à LACQRP. Le Q-learning déployé dans LACQRP et CRPLOGAQL est la technique RL majoritairement utilisée pour trouver le chemin de routage optimal dans les RCS. Cependant, pour les protocoles de routage RL centralisés avec un grand espace d'états et d'actions, le Q-learning de base utilisé pour mettre en œuvre ces protocoles

souffre d'une dégradation de la durée de vie du réseau et de la consommation d'énergie du réseau en raison du grand nombre d'épisodes d'apprentissage requis pour apprendre les TR optimales.

Enfin, dans cette thèse, une technique efficace basée sur RL sans modèle appelée Least-Square Policy Iteration (LSPI) est utilisée pour optimiser la durée de vie et la consommation d'énergie du réseau pour les RCS afin de surmonter la limitation de Q-learning. Le protocole conçu qui en résulte est appelé Protocole de routage centralisé pour l'optimisation de la durée de vie et de l'énergie avec GA et LSPI (CRPLEOGALSPI). Les résultats de la simulation montrent que le protocole CRPLEOGALSPI améliore la durée de vie et la consommation d'énergie du réseau par rapport au protocole CRPLOGAQL. En effet, CRPLEOGALSPI choisit une TR dans un état donné en considérant toutes les TR possibles et n'est pas sensible au taux d'apprentissage. De plus, alors que CRPLOGAQL évalue la politique optimale à partir des Q-valeurs, CRPLEOGALSPI met à jour les Q-valeurs en fonction des informations les plus récentes concernant la dynamique du réseau à l'aide de fonctions pondérées.

ABSTRACT

The sensor nodes' energy-efficient utilization is a major challenge in the design of Wireless Sensor Networks (WSNs). This is because the network lifetime is determined by the sensor nodes' limited energy sources whose replacement or recharging is almost impossible due to the mostly deployment of the sensor nodes in harsh environments. An effective way to prolong the network lifetime is by designing an energy-efficient routing protocol for WSNs using artificial intelligence such as Reinforcement Learning (RL) that can learn the network dynamics of WSNs. From the literature, the majority of the RL-based energy-efficient routing protocols for WSNs are distributed in nature. Though, the distributed RL-based routing protocol enables the wireless sensors to adaptively adjust to the dynamic changing nature of the WSNs environment which leads to reduced computational complexity and learning process time. However, the distributed RL-based routing protocols are limited to finding the global optimal routing paths. This leads to degradation in network lifetime and energy consumption. A centralized RL-based energy-efficient routing protocol can alleviate the challenge of finding the global optimal routing paths due to the global view of the WSNs. This thesis has three contributions which are presented in the sequel.

First, this thesis presents the design of a Lifetime-Aware Centralized Q-Routing Protocol (LACQRP) for WSNs to maximize the network lifetime. This is achieved by implementing Q-learning on the sink of the WSN, which also acts as a controller that has global knowledge of the network topology. The sink generates all possible distance-based Minimum Spanning Trees (MSTs) which form the set of Routing Tables (RTs) for a WSN with 100 sensor nodes. The maximization of the network lifetime is achieved by the controller learning the RT(s) that minimizes the maximum of the sensor nodes' consumption energies. The simulation results show that the LACQRP learns the optimal RT(s) that maximize the network lifetime and has a better convergence rate, network lifetime, and average energy consumption performance when compared with some distributed RL-based routing protocols for lifetime optimization, which are RL-Based Routing (RLBR) and RL for Lifetime Optimization (R2LTO). Although LACQRP extends the network lifetime, the computation time increases exponentially with the number of sensor nodes. Therefore, this method is impractical for large-scale WSNs.

Second, this thesis also proposes a method called Centralized Routing Protocol for Lifetime Optimization with GA and Q-learning (CRPLOGAQL) to reduce the computation time of LACQRP. This is achieved by replacing the All-MSTs algorithm of LACQRP with the proposed GA-based MSTs. Evaluation simulation shows that CRPLOGAQL can provide a suboptimal routing with reduced computation time when compared with LACQRP. Q-learning deployed in LACQRP and CRPLOGAQL is the majorly used RL technique to find the optimal routing path in WSNs. However, for the centralized RL-based routing protocols with large state space and action space, the baseline Q-learning used to implement these protocols suffers from degradation in the network lifetime and network energy consumption due to the large number of learning episodes required to learn the optimal RT(s).

Finally, in this thesis, an efficient model-free RL-based technique called Least-Square Policy Iteration (LSPI) is used to optimize the network lifetime and energy consumption for WSNs to overcome the limitation of Q-learning. The resulting designed protocol

is called a Centralized Routing Protocol for Lifetime and Energy Optimization with GA and LSPI (CRPLEOGALSPI). Simulation results show that the CRPLEOGALSPI has improved performance in network lifetime and network energy consumption when compared with CRPLOGAQL. This is because CRPLEOGALSPI chooses an RT in a given state considering all possible RTs and it's not sensitive to the learning rate. Also, while CRPLOGAQL evaluates the optimal policy from the Q-values, CRPLEOGALSPI updates the Q-values based on the most updated information regarding the network dynamics using weighted functions.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1 INTRODUCTION

1.1 BACKGROUND OF THE RESEARCH

A Wireless Sensor Network (WSN) can be defined as a collection of application-specific, low-powered, tiny devices called sensor nodes that are spatially deployed in a geographic area to monitor, collect, process, and cooperatively communicate real-time physical or environmental properties, such as temperature, sound, motion, pressure, humidity, etc. to a central device called the sink using the wireless medium [79, 114].

The advantages of WSN technology, when compared to traditional solutions of networking, are scalability, low costs, accuracy, reliability, flexibility, and deployment ease [118]. This has made WSNs efficient in different application fields, such as military, security, environment, and healthcare. However, the Quality of Service (QoS) requirements posed by these applications are limited by the resource constraints of the WSN, which are low power, short transmission range, low bandwidth, low memory, and limited processing and computing speed of the sensor nodes [100].

A wireless sensor node is a device that consists of a power unit, sensing unit, processing unit, and radio transceiver unit. The majority of the energy consumed in a sensor node is due to data communication with other sensor nodes [143]. The power unit consists of a limited energy source that supplies energy to the other units. Sensor nodes are mostly deployed in harsh environments, which makes sensor battery replacement difficult.

Subsequently, as the routing of data packets takes place between the sensor nodes and the sink, the energy gets reduced. Routing means finding the best possible routes from the sensor nodes to the sink [147]. A routing protocol is required between source sensor nodes which also act as routers in WSNs, to find the best paths between source nodes and the sink for reliable communication. Routing protocols are responsible for setting up paths for communication among sensor nodes and the sink [45].

Thus, routing in WSNs is an energy-consuming technique, which makes energy consumption and increasing network lifetime major challenges in WSNs [102]. This implies that if the path followed by a source sensor node to sink is not the best, more energy will be consumed. Energy-efficient routing protocols are expected to distribute the load among sensor nodes to reduce the energy consumption in WSNs and prolong the network lifetime [132].

Route optimization methods, therefore, play a vital role in WSNs, as optimal routing leads to less energy consumption and thus prolongs the network lifetime. Route optimization algorithms in WSNs consider multiple metrics, which include path length, energy, and network lifetime [68]. Thus resulting in a multi-objective optimization problem. Moreover, the dynamically changing topology of WSNs, resulting from sensor nodes stopping activities due to battery expiration from energy consumption, makes route optimization in WSNs a Non-deterministic Polynomial-time (NP)-hard problem [53].

This makes routing that uses traditional route optimization techniques based on a deterministic algorithm or dynamic programming such as Dijkstra and Floyd–Warshall not suitable for complex and highly changing conditions of WSNs. This is because of the huge assumptions regarding network condition changes and traffic flows [32]. Artificial intelligence, such as Reinforcement Learning (RL) and Genetic Algorithms (GA), can

be applied to find sub-optimal solutions by taking into consideration changing network conditions as they appear in practice [8, 125]. RL is a category of machine learning that solves a problem by learning with the trial-and-error method [31] while GA, is a type of evolutionary and search-based adaptive heuristic algorithm [109].

The energy-efficient utilization of sensor nodes can be achieved using three control techniques, which are decentralized control, distributed control, and centralized control [40]. In decentralized control, the nodes are divided into clusters. Each cluster has a central node that coordinates the activity of the nodes in each cluster. The activity of the nodes is therefore determined by the interaction of the central node of each cluster [33, 61]. For distributed control, each node makes local decisions with its partial knowledge of the entire network. This normally results in non-optimal routes in terms of energy consumption [104]. By contrast, in centralized control, the network's global knowledge is known by the sink using its centralized database. The sink carries out the routing decisions. The centralized method can lead to optimal routes [39].

The global knowledge of the sink/controller in the centralized control approach enables the use of optimal routing paths considering the constraint problem of energy-efficient routing in the WSN while maximizing the network lifetime [16]. In the centralized control approach, the routing and load-balancing decisions of the network are made by the sink since the sensor nodes have no intelligence. Therefore, the sensor nodes send data packets to the sink in a multi-hop manner using the routing path selected by the sink and stored as the sensor nodes' Routing Table (RT) [23]. An RT here is a Minimum Spanning Tree (MST) of the WSN graph. Therefore, the possible RT of the sensor nodes can be the possible MSTs generated by the sink after the network initialization [77].

However, the traditional centralized control approach for WSNs is limited by the sink using a predetermined routing path to receive data packets from the sensor nodes [122]. Since the predetermined routing path is selected, not taking into consideration the optimization problem of finding the best routing path(s) to balance the residual energy of sensor nodes during data transmission, the network lifetime is degraded. This is because the usage of the predetermined routing path does not consider the energy consumed by the sensor nodes to send packets to the sink [85]. This challenge of learning an energy-efficient way of selecting the best routing path(s) for the WSN's centralized control technique can be obtained by making the sink intelligent. One way of making the sink intelligent is to deploy artificial intelligence, such as RL, at the sink.

1.2 PROBLEM STATEMENT

Routing protocols are needed for energy and lifetime-efficient WSNs due to the limited power of wireless sensor batteries and deployment in harsh environments which make replacement difficult [116]. Intelligent routing protocols are required because of the dynamic and complex WSNs environment [20]. This has resulted in the use of RL algorithms to design energy-efficient routing protocols in WSNs so that the network lifetime is improved. The RL-based energy-efficient routing protocols find the optimal routing path over a certain time of the learning process to optimize the network lifetime [1]. From the literature, the majority of the RL-based energy-efficient routing protocols for WSNs are distributed in nature. The learning objects for the distributed RL-based routing protocols are the wireless sensor nodes which have the state information about the neighboring wireless sensors and take routing decisions depending on the defined reward function.

Though, the distributed RL-based routing protocol enables the wireless sensors to adaptively adjust to the dynamic changing nature of the WSNs environment which leads

to reduced computational complexity and learning process time. However, the distributed RL-based routing protocols are limited to finding the global optimal routing paths. This leads to degradation in network lifetime and energy consumption. A centralized RL-based energy-efficient routing protocol can alleviate the challenge of finding the global optimal routing paths due to the global view of the WSN. The sink of the WSN can act as the central controller with the learning agent. The sink requires all possible routing paths that connect with all sensor nodes without forming a loop (spanning trees) since the WSN can be represented as a graph. Centralized RL-based energy-efficient routing requires the use of MSTs. This is because the transmission energy of a sensor node is a function of link distance. Therefore, the sink can use the All-MSTs algorithm [115] to generate all possible routing paths with minimum total link distance. However, the problem of calculating all MSTs of a network graph is NP-hard. This makes the computational complexity of the All-MSTs algorithm vary exponentially with the number of sensor nodes in the WSN graph. This makes the most used baseline Q-learning for RL-based energy efficient protocols to require a long term of learning to find the optimal MST(s) that minimizes the network energy consumption while maximizing the network lifetime.

1.3 SIGNIFICANCE OF THE RESEARCH

The distributed RL-based routing protocols are limited to finding the global optimal routing paths. This leads to degradation in network lifetime and energy consumption [146]. Novel solutions for centralized RL-based energy-efficient routing protocol are designed to alleviate the challenge of finding the global optimal routing paths due to its global view of the WSNs.

The centralized control technique can use an All-MSTs algorithm to generate all the possible routing paths in the WSN. However, the problem of generating all the MSTs of a network graph is NP-hard. This makes the centralized control technique that uses the All-MST algorithm unfeasible in practice for very large WSNs. GA can be deployed at the sink to alleviate the NP-hardness associated with generating all the MSTs of large-scale WSN graphs using the All-MSTs algorithm. The centralized routing technique for lifetime optimization that uses GA enables the sink to generate a subset of MSTs for a large-scale WSN graph in polynomial time. The subset of all MSTs is then used as RTs by the sensor nodes to send data packets to the sink. Q-learning [7] is a kind of RL that can be used to learn the RT(s) that maximize the lifetime of the WSNs.

However, due to the large state space and action space as a result of the generated MSTs of the network graph, the baseline Q-learning can suffer from degradation in the convergence speed and network lifetime due to the large number of learning episodes required to learn the optimal routing path. Moreover, Q-learning is very sensitive to parameter settings; for example, changes in the learning rate affect the network lifetime. To overcome these limitations, a highly efficient model-free RL-based technique called Least-Squares Policy Iteration (LSPI) [8] can be used to replace Q-learning because LSPI chooses a routing path in a given state considering all possible routing paths. Also, LSPI is not sensitive to the learning rate. Moreover, while Q-learning evaluates the optimal policy from the Q-values, the LSPI updates the Q-values based on the most updated information regarding the network dynamics using weighted functions.

1.4 AIM AND OBJECTIVES

This research aims to design novel solutions for centralized RL-based routing protocols to maximize the network lifetime while minimizing the energy consumption in WSNs.

The research objectives are as follows:

(i) Performance analysis of some existing energy and lifetime aware distributed RL-based routing protocols to identify their strengths and weakness.

(ii) Design of lifetime-aware centralized RL-based routing protocol for WSNs.

(iii) Design of a centralized routing protocol for lifetime optimization using GA and Q-learning.

(iv) Design of a centralized routing protocol for a lifetime and energy optimization in WSNs using GA and LSPI.

1.5 SCOPE OF THE RESEARCH

The scope of this research is limited to RL-based methods for quality of service optimization in WSNs. The performance analysis of existing distributed RL-based energy-efficient routing protocols is carried out to provide the basis for the design of novel solutions for centralized RL-based energy-efficient protocols for WSN. The performance metrics considered in the research work include network lifetime, energy consumption, convergence rate, and simulation time. The WSN is modeled as a graph and performance analysis is carried out with simulations using the Python environment.

2

LITERATURE REVIEW

2.1 INTRODUCTION

This chapter provides a fundamental review of WSN and the mathematical techniques employed in this research work. These mathematical techniques include reinforcement learning, genetic algorithm, and graph theory. This chapter also entails a review of similar research work.

2.2 WIRELESS SENSOR NETWORK

A WSN can be defined as a collection of application-specific, low-powered, and tiny devices called sensor nodes that are spatially deployed in a geographical area to monitor, collect, process, and cooperatively communicate real-time physical and/or environmental properties such as temperature, sound, motion, pressure, humidity, etc. to a central device called a sink using the wireless medium [114]. The collected data is sent to the sink (local user) using multi-hop transmissions. Subsequently, the sink sends the data to a remote user through a gateway known as the base station connected to the internet as shown in Figure 2.1.

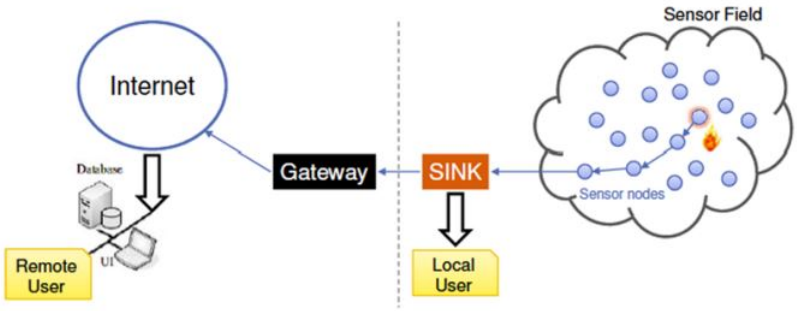

Figure 2.1: A WSN [118]

The advantages of WSN technology, when compared to traditional solutions of networking, are scalability, low costs, accuracy, reliability, flexibility, and deployment ease [118]. This has made WSN be used in different application fields such as military, security, environment, and healthcare. Sensor nodes are used in the military for battlefield monitoring. In the case of the environment, sensor nodes can be used to forcast weather changes and natural disasters ahead of time. For security, sensor nodes can be used for surveillance and to provide alertness to terrorist attacks. While, for health, sensor nodes can be used to monitor a patient's health condition [77]. However, the QoS requirements posed by these applications are limited by the resource constraints of the WSN which are low power, short transmission range, low bandwidth, low memory, and limited processing and computing speed of the sensor nodes [100].

2.2.1 *Architecture of Sensor Network*

A WSN is made up of wireless sensor nodes. Basically, a wireless sensor node is a device that consists of the power unit, sensing unit, processing unit, and radio transceiver unit as shown in Figure 2.2.

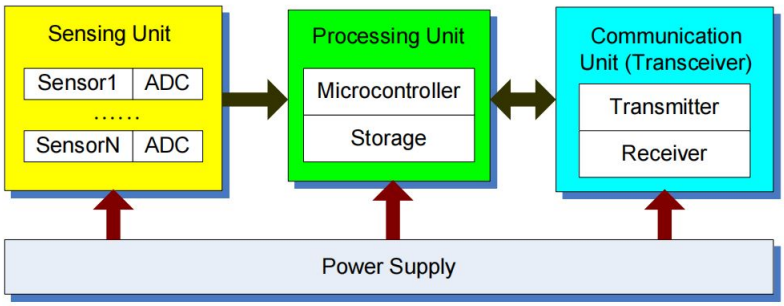

Figure 2.2: Architecture of Wireless Sensor [143]

The sensing unit consists of one or more sensors. Each sensor is connected to an Analog-to-Digital-Converter that digitized the analog signals from a sensor and sends the digital signal to the processing unit. The processing unit tasks include processing data and coordinating the functions of other units of the sensor. The communication unit connects the sensor node with another through the wireless medium and the power unit supplies energy to the other units. The majority of the energy consumed in a sensor node is from data communication with other sensor nodes [45]. The power unit consists of a limited energy source that supplies energy to the other units. Consequently, the sensor nodes are mostly deployed in harsh environments which makes sensor battery replacement difficult. This makes the energy-efficient utilization of the sensor nodes vital in order to prolong the network lifetime [72].

2.2.2 *WSN Structure and Control*

The sensor field to be monitored play a vital role in finding the network topology size and deployment. WSNs are usually deployed in two ways which are structured deployments and unstructured deployments. In structured deployment, the sensor nodes are deterministically placed in predefined locations. Structured deployment is easy to manage because it contains few sensor nodes deployed in a small indoor deployment area. Structure deployment is also used in large outdoor area environment that requires many sensor nodes. For Unstructured deployments, many sensor nodes are deployed in an ad-hoc manner. Unstructured deployment is used if the sensor field is large and inaccessible by the network designers. The resulting WSN from unstructured deployment is difficult to manage. Subsequently, WSNs can be controlled in three ways which are centralized control, decentralized control, and distributed control [102].

(i) Centralized control: The network's global knowledge is known by the sink by means of its centralized database. The sink carries out the routing decisions. The centralized method can lead to optimal routes.

(ii) Decentralized control: The nodes of the network are divided into clusters. Each cluster has a central node that coordinates the activity of the nodes in each cluster.

The activity of the nodes is therefore determined by the interaction of the central node of each cluster.

(iii) Distributed control: Each node takes local decisions with its partial knowledge about the entire network. This normally results in non-optimal routes in terms of energy consumption.

2.2.3 *WSN Communication Standard*

IEEE 802.15.4 protocol is the communication standard that supports WSN design requirements of low bandwidth, low complexity, low-cost implementation, short transmission range, and low power consumption. The bands of operation of the physical layer of IEEE 802.15.4 are between (0.868/0.915 - 2.4) GHz. Both commercial and academic wireless sensor nodes support IEEE 802.15.4 protocol [85].

Two types of communication can be defined for WSNs. These are single-hop communication and multi-hop communication. In single-hop communication, the transmission range of all sensor nodes in the network is adequate to send the monitored data directly to the sink as shown in Figure 2.3.

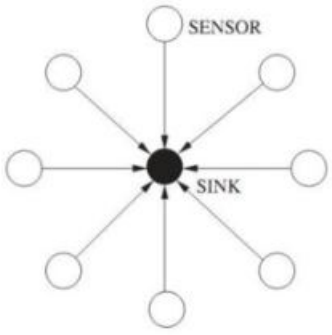

Figure 2.3: Single-Hop Communication

However, this leads to an increase in energy consumption of the sensor nodes and subsequent depletion of the energy source of the sensor nodes especially when the sensor nodes are deployed in a large geographical area. Since sensor nodes are deployed in a large geographical area, the need to reduce the energy consumption of sensor nodes and consequently increase the lifetime of the WSNs leads to the frequent use of multi-hop communication.

In multi-hop communication, sensor nodes collaborate together to send the monitored data to the sink as shown in Figure 2.4.

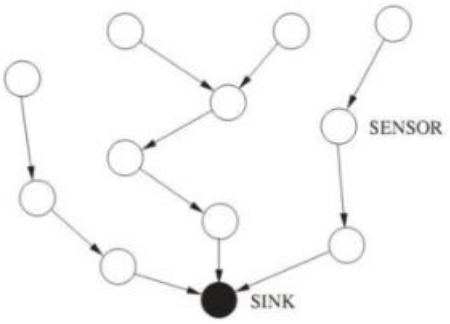

Figure 2.4: Multi-Hop Communication

This results in the routing problem which involves finding the optimal multi-hop path for a sensor node to send the monitored and captured data to the sink [40].

2.2.4 *Challenges in WSNs*

The design and deployment of WSNs are mostly dependent on the application intended. The different challenges in WSNs are energy consumption, QoS, adaptability, localization, security, and privacy. Therefore is vital to find a trade-off solution for these challenges in designing and deploying WSNs [40].

2.2.4.1 *Energy Consumption*

Different applications in WSNs have different power level requirements. The reduction of the energy consumption of the sensor nodes is one of the most important challenges in the design of WSNs [12]. This is because the lifetime of the sensor nodes is determined by the limited energy of the sensor nodes and the rate of energy dissipation. Also, in many cases, replacing or recharging the sensor nodes is almost impossible. An effective way to prolong the lifetime of the sensor node is by designing an energy-efficient routing protocol for WSNs. This is because more energy is consumed by the sensor nodes if the routing path used in sending data to the sink is not the best [31].

The lack of battery replacement, which is essential for affordable WSN deployment, requires energy-efficient operations. Since high reliability and low delay may demand a significant energy consumption of the network, thus reducing the WSN lifetime, the reliability and delay must be flexible design parameters that need to be adequate for the requirements. Controllers can usually tolerate a certain degree of packet losses and delay. Hence, the maximization of the reliability and minimization of the delay is not the optimal design strategies for the control applications [67].

2.2.4.2 *Quality of Service*

There are two perspectives on QoS in WSNs. These are application-specific and network-specific. For the application-specific QoS, there are mainly two metrics used to measure the QoS which are latency and data reliability. Data reliability is the measure of the accuracy of the data received by the sink. Meanwhile, Sensor information must be sent to the sink of the network with a given probability of success because missing this data could prevent the correct execution of control actions or decisions concerning the phenomena sensed. However, maximizing the reliability may increase substantially the

network energy consumption. Hence, network designers need to consider the trade-off between reliability and energy consumption [19].

Latency is the measure of the end-to-end data packet delivery delay. Sensor information must reach the sink within some deadline. A probabilistic delay requirement must be considered instead of using average packet delay since the delay jitter can be too difficult to compensate for, especially if the delay variability is large. Re-transmission of old data to maximize reliability may increase the delay and is generally not useful for control applications [46].

Subsequently, the network-specific QoS gives a measure of how the WSNs can meet the application-specific QoS while efficiently utilizing network resources of bandwidth and energy of sensor nodes. Examples of parameters used to measure network-specific QoS are network throughput and lifetime [115].

2.2.4.3 *Adaptability*

WSNs should be designed taking consideration flexibility, scalability, fault tolerance, and self-management in mind. Flexibility implies that a WSN can support many application scenarios. Some of the applications require that more nodes are added to the network. The Scalability of a WSN implies that the network can scale with the number of sensor nodes. Consequently, since once WSNs are deployed, it is expected that the network function with little or no human intervention, WSNs should have fault tolerance and self-management ability [67].

That is due to the link failure inherent in the wireless medium and the limited energy of the sensor nodes, a WSN should be robust and intelligent to take into consideration the dynamic changes of the network topology [40]. Also, Since the processing resources are limited, the protocol procedures must be computationally light. These operations should be performed within the network to avoid the burden of too much communication with a central coordinator. This is particularly important for large networks. The protocol should also be able to adapt to size variation of the network, for example, caused by moving obstacles, or the addition of new nodes.

The network operation should adapt to application requirement changes, varying wireless channels, and network topology. For instance, the set of control application requirements may change dynamically and the communication protocol must adapt its design parameters according to the specific requests of the control actions. To support these changing requirements, it is essential to have an analytical model describing the relation between the protocol parameters and performance indicators (reliability, delay, and energy consumption) [9].

2.2.4.4 *Security and Privacy*

WSNs are prone to security attackers due to the wireless medium and the multi-hop transmissions. The two main security issues suffered by WSNs are node authentication and privacy preservation. Privacy means that data confidentiality is preserved using robust and intelligent security mechanisms during the routing of the data from the source to the destination. Node authentication involved the use of a well-structured mechanism to ensure that malicious nodes those not have access to the data transmitted in the WSNs [93, 109].

2.2.5 *WSNs Applications*

WSNs can contain different kinds of sensors: seismic, sonars, biomedical, magnetic, thermal, visual, infrared, acoustic, radar e.t.c which can monitor, track or detect a wide variety of ambient conditions like temperature, pressure, humidity, direction, speed, movement, soil makeup, light, noise levels, the absence or presence of certainty of objects, and mechanical stress levels on attached objects. This has made a wide range of WSN application to be possible [17, 60, 91].

This wide range of applications includes solar system climate exploration, weather analysis, and prediction, monitoring of the environment, battlefield surveillance and monitoring, homeland security, monitoring of space assets for potential and human-made threats in space, ground-based monitoring of both land and water, intelligence gathering for defense, urban warfare, and beyond, monitoring seismic acceleration, temperature, strain, Global Positioning System (GPS) data, and wind speed [128].

The ever-increasing applications of WSNs can be mainly grouped into five classes which are military, environmental, health, home, and industrial applications. These applications are explained in the sequel.

2.2.5.1 *Military Applications*

The first field of human endeavors that applied WSNs is the military which has resulted in the motivation of research in WSNs. The earliest research effort was carried out in the late 90s and is known as smart dust which involves the development of very small size sensor nodes for spying activities in hostile environments [65, 139].

WSNs are an integral part of military command, communications, control, computing, surveillance, intelligence, reconnaissance, and targeting systems. The rapid deployment, fault tolerance, and self-organization characteristics of WSNs make sensing techniques for the military very promising. Because WSNs are based on the dense deployment of low energy consumption low-cost and disposable sensor nodes, the damage of some sensor nodes by hostile action does not interrupt military operations when compared to the damage of a traditional sensor. This makes the WSN concept suitable for battlefields. Some of the military applications of WSNs are monitoring equipment and ammunition; combat monitoring; intruder detection; battlefield surveillance; reconnaissance of opposing forces and terrain; targeting; battle damage assessment; and biological, chemical, and nuclear, attack reconnaissance and detection [6, 22, 151].

2.2.5.2 *Environmental Applications*

The independent coordination abilities of WSNs are used in the attainment of a large diversity of environmental applications. Part of these environmental applications of WSNs are movement tracking of small animals, birds, and insects; monitoring environmental situations that impact livestock and crop; irrigation; planetary exploration and earth monitoring; biological/chemical detection; precision agriculture; biological, earth, and environmental monitoring in soil, marine, and atmospheric contexts; forest fire detection; geophysical or meteorological research; flood detection; and pollution studies [47].

2.2.5.3 *Health Applications*

WSNs find usage for biomedical applications because of the rapid developments of advanced smart integrated medical sensors and biomedical implanted devices. Examples

of health applications for WSNs are the administration of drugs in hospitals; monitoring and tracking of patients and doctors in the hospital; diagnostics; patient integrated monitoring; internal processes and movements monitoring of small animals like insects; interfaces provision for the disabled; and human physiological data telemonitoring [43, 80, 97].

2.2.5.4 *Industrial Applications*

Industrial fields have long used wired sensor networks for industrial access control, building automation, sensing, and control functions. However, the wired sensor networks are limited because the cost of upgrading the sensor system of an industrial plant is as deploying a new system. Industrial applications have also utilized manual monitoring for preventive maintenance. While wired sensor networks have a high cost of deployment, manual systems monitoring requires personnel and has limited accuracy. WSNs have been used to reduce these limitations because of their high granularity, deployment ease, and high accuracy provided by wireless communication stations powered by batteries [11].

Examples of industrial applications are material fatigue monitoring; virtual keyboards building; inventory management; product quality monitoring; constructing smart office spaces; office buildings environmental control; automatic manufacturing environments guidance and robot control; interactive museums; interactive toys; automation and control of factory process; disaster areas monitoring; smart structures with embedded sensor nodes; diagnosis of a machine; factory instrumentation; transportation; actuators local control; monitoring and detecting car theft; tracking and detection of a vehicle; rotating machinery, anechoic chambers, and wind tunnels; and realization of cognitive radio networks distributed spectrum sensing [3]

2.2.5.5 *Urban Applications*

WSNs offered different kinds of sensing capacities that provide the ability to acquire about a given area, be it a building, a room, or outdoors an unrivaled amount of information. This makes the temporal and spatial features of any event in an urban environment to be measured by WSNs thereby providing countless amount of applications. Transportation systems, monitoring of structural health, smart cities, and smart homes are a few examples of urban applications of WSNs [117].

In large cities monitoring various parameters is crucial for the optimal life of the citizens. Smart cities entail monitoring different parameters like the number of vehicles moving in a particular direction, and real-time provision of traffic data to authorities in large cities to make life optimal for citizens. Therefore, applications like indicating car parking spots, monitoring traffic levels, etc are achieved by WSNs [58, 74, 133].

As WSN technology forge ahead, it became possible for the actualization of smart homes which involves embedding smart sensor nodes and actuators in home appliances like Digital Versatile Disc (DVD) players, vacuum cleaners, refrigerators, microwave ovens, and also water monitoring systems. These sensor nodes inside the home appliances can interact with an external network and each other through satellite or the internet. This has enabled end-users to manage home appliances remotely or/and locally quite easily. Therefore, the use of WSNs at home has enabled the interconnection of various appliances with convenient control. For example, wireless sensors read the utility meters at homes remotely and send the read data to a remote center using the internet or satellite [112, 135].

2.2.5.6 *Discussion*

The operation of WSNs for all applications faces general issues, due to challenges in wireless communication and resource constraints of the sensor nodes. Particularly, WSN sensor nodes are limited because of severe energy constraints. This is because the energy sources are provided by batteries which are difficult or impractical to be either replaced or recharged since sensor nodes are usually deployed in difficult or impossible-to-reach locations. Therefore, the achievement of energy conservation is a crucial problem for WSNs. The different WSNs applications need different specifications of requirements for their realization in real-world applications.

For example in military applications, the sensor nodes are scattered in the field to monitor and protect military personnel from being harmed by intruders. In this situation changing or charging the sensor nodes' batteries is difficult or not possible, therefore the sensor nodes must be power efficient since sensor nodes in this application cannot be put to hibernate or sleep mode. Likewise, the sensor nodes use to collect data should operate responsively without failure. This implies that in case of a failure in any sensor node, the other sensor node should work independently to keep the WSN active. Likewise, the sensor nodes used in the military application should be designed to consume low power, yielding to increasing the efficacy and reliability of the WSN. Also, the routing protocols used on the network should be scalable, and adaptable to changes in the size of the sensor nodes and traffic load to provide the required throughput. This implies that WSNs for military applications should meet optimal throughput, security, reliability, and resistance to intervention and jamming.

For health applications basically use for the purpose of monitoring patients, the sensor nodes have to work for a longer period. This implies that the sensor nodes and routing protocol should be power efficient. Sensor nodes with longer life can monitor the state of a patient effectively. For example blood pressure monitoring system for blood pressure state and a diabetes monitoring system for sugar level. This requirement is crucial because often the sensor nodes are implanted in the patient's body and cannot be put in sleep mode to enable continuous monitoring and transmission. The sensor nodes have to be responsive, autonomous, and reliable to collect and transmit the data regularly with very high reliability. Scalability and adaptability to changes in the network environment, while accurate information is being collected, are very crucial.

In environmental applications, communication should withstand jamming because emergency alerts should be transmitted without delay. Also, the WSN should be operational despite the loss of a certain number of nodes.

For industrial applications, most functions are time-critical and there is a remarkable presence of interference due to electromechanical signals. Therefore, communication should yield optimal throughput, reliability, and resistance to interference and jamming. There is also a need for the operations to be carried out under firm security standards.

Urban applications which are classified into indoor applications and outdoor applications need different specification requirements. For indoor applications, WSN should yield firm security for privacy protection and withstand interference generated by other home devices. Consequently, for outdoor applications, the communication should provide high throughput levels, reliability, security, and resistance to intervention and jamming because of the large amount of transmitted data. Also, the WSN should withstand the loss of a certain number of nodes.

The summary of the specification requirements for the different WSN applications mentioned is provided in Table 2.1.

Table 2.1: Summary of specification requirements for WSN applications

Application Type		Specification Requirements				
		Energy	**Network Tolerance**	**Reliability**	**Throughput**	**Security and Privacy**
Military		Very High	Very High	Very High	Very High	Very High
Environmental		Very High	Very High	High	Very High	High
Health		Very High	High	Very High	Very High	High
Industrial		Very High	Very High	Very High	Very High	High
Urban	Indoor	Very High	High	Very High	Medium	Very High
	Outdoor	Very High	Very High	Very High	Very High	Very High

2.2.6 *Protocol Architecture of WSNs*

The protocol architecture provides the standardization of the interaction among sensor nodes and sinks thereby enabling the development of compatible products (hardware and/or software) by various vendors. The protocol stack enables the combination of energy-efficient routing awareness management, integration of data networking protocols, efficient power communication on the wireless link, and the promotion of cooperation among sensor nodes [100]. The WSN protocol stack as shown in Figure 2.5 is made up of the physical, data link, network, transport, and application layer.

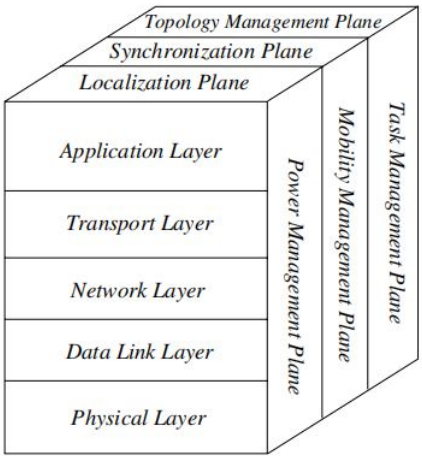

Figure 2.5: WSN Protocol Stack [7]

The protocol stack also consists of synchronization, localization, topology management, power management, mobility management, and task management plane.

(i) Physical layer: This layer enables data modulation, encryption, transmission, and reception techniques.

(ii) Data link layer: This is in charge of ensuring good communication using data stream multiplexing, frame detection, error control methods, and enabling channel access through the Media Access Control (MAC) to reduce collision with neighbors' broadcasts and provide power-aware communication as a result of the noisy nature of the wireless media and the changing topology of WSNs.

(iii) Network layer: This is responsible for data routing using wireless multi-hop routing protocols among sensor nodes and sink.

(iv) Transport layer: This is in charge of maintaining data flow if the wireless sensor network application needs it and is linked to the internet.

(v) Application layer: This enables the building of different types of application software depending on the sensor nodes sensing tasks. This layer enables the software and hardware of the lowest layer explicit to the end users.

(vi) Task, mobility, and Power management levels: These are in control of controlling the task distribution, node movements, and consumed energy across the whole protocol stacks. These planes enable sensors to organize their tasks and collaborate in an energy-efficient manner thereby prolonging the lifetime of the sensor nodes and sharing.

2.2.7 *Routing in WSNs*

Routing means finding the best possible routes from sensor nodes to the sink. A routing protocol is required between source sensor nodes acting as routers in WSNs to identify the best paths between source nodes to the sink for reliable communication to occur. Routing protocols are responsible for setting up paths for communication to occur among sensor nodes and the sink. Routing protocols are categorized as the flat routing protocol also known as the data-centric routing protocol and the hierarchical routing protocol [8].

In a flat routing protocol, all sensor nodes have a special global address and are at the same level. Examples of flat routing protocols are flooding, gossiping, and direct diffusion. Subsequently, in a hierarchical routing protocol, the sensor nodes of the WSNs are divided into clusters forming a hierarchy. Examples of hierarchical routing protocols are low-energy adaptive clustering hierarchy (LEACH), threshold-sensitive energy-efficient sensor network (TEEN), power-efficient gathering in sensor information systems (PEGASIS), and adaptive threshold-sensitive energy-efficient sensor network (APTEEN) [7]. In WSNs, every sensor node is initially preinstalled with a battery that has a limited amount of energy and can't be easily replaced. But, as the routing of data packets takes place between the sensor nodes and the sink, the energy gets reduced. Thus, routing in WSNs is an energy-consuming technique, which makes reducing energy consumption and increasing network lifetime major challenging problems in a WSN [106].

This implies that if the path followed by a source sensor node to sink is not the best, more energy will be consumed. Energy-efficient routing protocols are expected to distribute the load among sensor nodes to reduce the energy consumption in WSNs and prolong the network lifetime [13]. Therefore, special care must be considered in designing routing protocols for WSNs. Route optimization methods, therefore, play a vital role in a WSN, as optimal routing will lead to less energy consumption and thus prolong the network lifetime. Route optimization algorithms in WSNs consider multiple metrics which are path length, energy, and network lifetime which results in a multi-objective optimization problem.

Also, the dynamic changing topology of WSNs as a result of sensor nodes stopping activities due to battery expiration from energy consumption makes route optimization in WSNs an NP-hard problem [98]. This makes traditional routing that uses traditional route optimization techniques based on a deterministic algorithm or dynamic programming such as Dijkstra and Floyd-Warshall unsuitable for complex and highly changing conditions of WSNs. This is because of the huge assumptions regarding network condition changes and traffic flows. Artificial intelligence such as RL and GA can be applied to

get sub-optimal solutions by considering changing network conditions as they appear in practice [8].

2.2.7.1 *Routing Problem in WSNs*

Practical WSNs are not fully connected. This means that all sensor nodes are not directly connected to each other due to the limited power resources and transmission range. If each sensor node is associated with a unique identification index (ID), the set of sensor nodes in the WSN can be defined as $\mathcal{S} = \{1, 2, ..., n\}$, with the sink of the WSN being node 1.

All the nodes that a node i can communicate directly with are those within its communication range. This set of nodes is referred to as the neighbor of node i and it is represented as $\mathcal{N}_i$, $i \in \mathcal{S}$. Modeling the WSN as an undirected graph implies that $i \in \mathcal{N}_j \iff j \in \mathcal{N}_i$.

This means that for nodes that are not directly connected to communicate, packets will have to hop between nodes along the paths between these nodes resulting in multi-hop routing. Therefore, a path $\mathcal{P}$ from sensor node i to the sink is a sequence of pairwise neighboring sensor nodes that begins at sensor node i and terminates at node 1, the sink of the WSN. That is $\mathcal{P} = \{p_0, p_1, ..., p_n\}$, where p_0 is the source sensor node i, p_n is the sink, node 1, and $p_i \in \mathcal{N}_{i-1}$.

The routing algorithms that are required to find paths between non-neighboring nodes need some kind of metric to enable the generation of a good path when comparing different paths. A routing algorithm generates a routing path by comparing the links between sensor node i and its neighboring nodes and assigning a cost required for communication between two neighboring nodes.

The cost of sending packets between two neighboring nodes i and j at time t is represented by a cost function given as $c(i, j, t)$, $j \in \mathcal{N}_i$. The time variable implies that the cost of a link varies over time. This allows the definition of the cost function for an entire path at time t which can be given as:

$$c(\mathcal{P}, t) = \sum_{p_i \in \mathcal{P}} c(p_i, p_{i+1}, t) \tag{2.1}$$

2.3 REINFORCEMENT LEARNING

RL is a field of machine learning that enables an agent to learn the dynamic behavior of its environment by taking an action depending on its current state which improves the learning with time (maximizing the concepts of cumulative reward) using trial and error interaction on the environment [131]. For example, a sink interacts with all the sensor nodes in the network to make routing decisions for the sensor nodes. In this case, the agent is the controller, the environment is the controller's neighborhood, the state is the current multicast tree the nodes are using to send data packets, and the action is the selection of the next unicast tree to be used by all the nodes to send packets. An RL problem is solved by modeling the problem as a Markov Decision Process (MDP) [120] as shown in Figure 2.6.

The MDP consists of four tuples (S, A, P, R), where S denotes the set of states, A is the set of actions, P is the matrix of state transition probability and R is the reward function. The combination of P and R is used to describe the model of the environment. The probability to be in a given state $S_{t+1} = \hat{s}$ from a current state $S_t = s$ by taking action

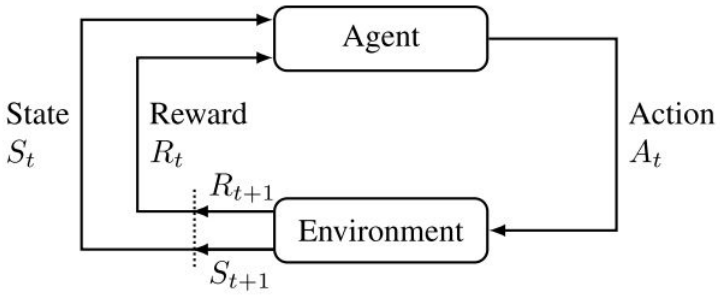

Figure 2.6: RL Model as an MDP [131]

$A_t = a$, and getting a reward $R_{t+1} = r$ is known as the transitional probability given in Equation 2.2 [76]:

$$p(\hat{s}, r|s, a) = Pr(S_{t+1} = \hat{s}, R_{t+1} = r|S_t = s, A_t = a) \tag{2.2}$$

The reward function enables the environment to provide feedback as a form of reward to the agent. The reward is the measurement of the effect of the recently taken action by the agent from its current state.

The two methods of RL problems are model-free and model-based methods. In the model-free method, the agent enhances its policy without inferential knowledge of the model of the environment. That is the matrix of the state transition probabilities is not needed. Whereas, in the model-based method, the agent learns the model of the environment by computing the matrix of the state transition probabilities and then enhances its policy to approach optimality.

The model-based methods learn faster than the model-free methods because the information stored in their internal model is reused. However, the model-based methods are not mostly used in practice due to their dependency on the initial environmental model accuracy and larger size of storage cost and computations. Consequently, in both methods, the role of the agent is to maximize a discounted global reward received over time while finding a policy, that map states to actions.

A policy π_t determines the behavior of learning of an agent at a given instance of time t. In RL, what is bad or good in an immediate sense is depicted by a reward function. While what is bad or good in the long run is depicted by a value function. There are two types of Q-value functions which are the action-value function and the state-value functions. The action-value function computes how good it is for an agent in a given state to take a given action. While the state-value function computes how good it is for an agent to be in a given state.

The majority of the work carried out on RL to solve unicast routing problems in networks used model-free methods. This is because they don't need the network's environmental models which are difficult to get as a result of their dynamically changing properties like the residual battery capacities, lifetime of nodes, etc. in WSN.

In RL, the quality of taking an action, $A_t = a$ from a state, $S_t = s$ is given by a Q-value, $Q(s, a)$ which is known as the state-action value function. The aim of finding a solution to an MDP is to obtain an optimal policy, which maps states to actions in a view to maximizing the cumulative reward. The aim of finding a solution to an MDP is the same as finding the Q-fixed points given by the Bellman equation as given in Equation 2.3.

$$Q^{\pi}(s,a) = r(s,a) + \gamma \sum_{\hat{s}} p(\hat{s}|s,a) \max_{\hat{a}} \{Q^{\pi}(\hat{s},\hat{a})\} \tag{2.3}$$

where $Q^{\pi}(s,a)$ is the expected, discounted total reward when taking action a at state s, and thereafter following policy π, $r(s,a)$ is the expected reward gotten immediately by taking an action a when in a state s, and it is calculated as in Equation 2.4:

$$r(s,a) = \sum_{\hat{s}} p(\hat{s}|s,a) R(s,a,\hat{s}) \tag{2.4}$$

where $p(s|s,a)$ is the probability transition model that gives the probability of being in a state s after taking an action a when in state s. $R(s|s,a)$ is the immediate reward gotten when an agent takes an action a when at state s and transit to state $\hat{s}$. The second term in Equation 2.4 is the expected maximum future reward. γ is the discount factor and models the fact that the future reward is less valuable than the immediate reward.

In the context of a centralized learning routing technique, each minimum spanning tree (MST) is a state s, and for each of the other MSTs which is a possible next state $\hat{s}$, is an action a with transition probability $p(\hat{s}|s,a) = 1$. Taking action a, at state s means the sink sends the selected MST as the routing table to the sensor nodes for data transmission to the sink. This enables getting the optimal routing path from a succession of table look-up processes. This challenge of learning the optimal routing policy is similar to solving the Bellman equation given in Equation 2.3

2.3.1 *Q-Learning*

Watkins, (1989) proposed a model-free learning technique called Q-learning to solve the Bellman equation deterministically when the state probability and reward system is known [140]. Q-learning is an off-policy temporal-difference control algorithm, that enables the direct approximation of the Bellman equation. Q-learning has been mostly used for solving unicast routing RL problems in networks. This is because the technique was proven to converge to optimum action values with one as the probability if and only if all actions are continuously sampled in all of the states.

In this technique, the quality of being in a state $s \in S$ and choosing an action $a \in A$ is measured by an action-value function called Q-value. The Q-value is a measure of the long-run reward that the agent gets from each pair of state-action. The estimate of this action-value function $q(s,a)$ which is used to find the best action for a given state is realized by caching the Q-values $Q(S_t, A_t)$ of pairs of state-action using the iterative update rule given in Equation 2.5.

$$Q(S_t, A_t) = (1-\alpha)Q(S_t, A_t) + \alpha \left[R_{t+1} + \gamma * max\{Q(S_{t+1}, a)\} \right] \tag{2.5}$$

Where $0 < \alpha \leq 1$ is the learning factor and $0 \leq \gamma \leq 1$ is the discount factor. The extent to which the newly learned Q-value affects the old Q-value is dependent on the learning factor. The closer the value of α is to one, the more the impact of the newly computed Q-value on the old one. If is equal to one, then the recently learned Q-value replaces the old Q-value completely. The discount factor controls the agent's liking for future rewards with respect to the current reward. If γ is equal to 1, both the immediate reward and the future reward are considered equally.

Q-learning being a model-free RL method is suitable for learning optimal routing strategy in WSNs because of its simplicity and non-requirement of any knowledge of the underlying transition and reward mechanism. However, Q-learning suffers the following drawbacks when used in learning optimal routing strategies in WSNs.

(i) A large number of iterations are required to learn the optimal routing path, this leads to the degradation of the convergence speed and routing performance.

(ii) It is very sensitive to parameter settings, for example, changes in the learning rate affect the routing performance.

2.3.2 *Least-Squares Policy Iteration*

Least-Squares Policy Iteration (LSPI) is a model-free, off-line, and off-policy approximation policy iteration RL technique proposed by Lagoudakis and Parr (2003) [89]. LSPI addresses the challenges associated with Q-learning by replacing the direct evaluation of the optimal state-action value function of the Bellman equation by approximating Q-values for each policy using a linear weighted function approximator. That is, given a set consisting of k state-action-dependent basic functions $\varphi(s,a)$ that provide the information of the selected state-action pair features as given in Equation 2.6.

$$\mathcal{F} = \left\{ \varphi_j(s,a) : S \times A \mapsto R,\ j = 1,...,k \right\} \tag{2.6}$$

The basic functions are fixed and manually designed. The state-action value function is therefore approximated as the weighted linear combination of the k basic functions as given in Equation 2.7.

$$\hat{Q}^{\pi}(s,a) = \sum_{j=1}^{k} \varphi_j(s,a) w_j = \varphi(s,a)^T w \tag{2.7}$$

where w_j is the weight associated with the j^{th} basic function. From the approximated form of Equation 2.3 using Equation 2.7, the matrix form of Equation 2.3 can be written as in Equation 2.8.

$$\Psi w \approx R + \gamma P^{\pi} \Psi w \tag{2.8}$$

where Ψ is a matrix of basic functions for each state-action pair and is of size $|S||A| \times k$. Equation 2.8 can be reformulated as Equation 2.9 for a linearly dependent columns of Ψ.

$$\Psi^T (\Psi - \gamma P^{\pi} \Psi) w^{\pi} = \Psi^T R \tag{2.9}$$

Solving the linear system in Equation 2.9 leads to the extraction of the weights associated with $\hat{Q}^{\pi}(s,a)$ in Equation 2.7. The equation for extracting the weights can be written as in Equation 2.10.

$$w^{\pi} = X^{-1} y \tag{2.10}$$

where X and y are given in Equation 2.11 and Equation 2.12, respectively.

$$X = \Psi^T(\Psi - \gamma P^\pi \Psi) \tag{2.11}$$

$$y = \Psi^T R \tag{2.12}$$

LSPI being a model-free off-policy learning algorithm learns X and y using sampling from the environment. Subsequently, the learned X and y are used to learn the weights to approximate the state-value function Q^π of a fixed policy π from the obtained samples using the Least-Squares Temporal-Difference Learning (LSTDQ) [89]. The LSTDQ is an algorithm similar to the Least-Squares Temporal-Difference (LSTD) learning algorithm [26] and learns the approximate state-action value function of a fixed policy, therefore allowing action selection and policy improvement without a model.

Therefore with a set of samples $\mathcal{D} = \left\{(s_i, a_i, r_i, \hat{s_i}) | i = 1, 2..., M\right\}$ gotten from the environment, the approximated version of Ψ, $P^\pi\Psi$, and R is constructed using Equations 2.13, 2.14, and 2.15, respectively.

$$\hat{\Psi} = \begin{pmatrix} \Psi(s_1, a_1)^T \\ \vdots \\ \Psi(s_i, a_i)^T \\ \vdots \\ \Psi(s_M, a_M)^T \end{pmatrix} \tag{2.13}$$

$$P^{\hat{\pi}}\Psi = \begin{pmatrix} \Psi(\hat{s_1}, \pi(\hat{s_1}))^T \\ \vdots \\ \Psi(\hat{s_i}, \pi(\hat{s_i}))^T \\ \vdots \\ \Psi(\hat{s_M}, \pi(\hat{s_M}))^T \end{pmatrix} \tag{2.14}$$

$$\hat{R} = \begin{pmatrix} r_1 \\ \vdots \\ r_i \\ \vdots \\ r_M \end{pmatrix} \tag{2.15}$$

Therefore, the approximated $\hat{X}$ and $\hat{y}$ can be given as in Equation 2.16 and 2.17, respectively.

$$\hat{X} = \frac{\hat{\Psi}^T(\hat{\Psi} - \gamma P^{\hat{\pi}}\hat{\Psi})}{M} \tag{2.16}$$

$$\hat{y} = \frac{\hat{\Psi}^T \hat{R}}{M} \tag{2.17}$$

Since in a practical evaluation of $\hat{X}$ and $\hat{y}$, M is finite, therefore the solution to the system will not be affected if the factor $1/m$ is dropped. If $\hat{X}$ and $\hat{y}$ combined can be gotten in a single sample, then constructing an iteration update rule for $\hat{X}$ and $\hat{y}$ is feasible.

Assuming $\hat{X}^0 = 0$ and $\hat{y}^0 = 0$ initially, the current learned approximates of X and y for a fixed policy π, will be $\hat{X}^t$ and $\hat{y}^t$, respectively. Therefore, Equation 2.18 and Equation 2.19 will give the approximated values of $\hat{X}^{t+1}$ and $\hat{y}^{t+1}$, respectively of a new sample $(s_t, a_t, r_t, \hat{s_t})$.

$$\hat{X}^{t+1} = \hat{X}^t + \varphi(s_t, a_t)\Big[\varphi(s_t, a_t) - \gamma\varphi(\hat{s_t}, \pi(\hat{s_t}))\Big]^T \tag{2.18}$$

$$\hat{y}^{t+1} = \hat{y}^t + \varphi(s_t, a_t) r_t \tag{2.19}$$

The weight is updated as the iteration procedure repeats with the improved policy until the optimal policy is reached. That is the weights of policies between successive iterations do not differ significantly. Therefore, the learning agent in a given state chooses its action in each learning round using the learned policy given in Equation 2.20.

$$\pi(s|w_\pi) = \underset{a}{\text{argmax}}(\varphi(s, a)^T w_\pi) \tag{2.20}$$

2.3.3 *Exploitation versus Exploration*

Reinforcement learning requires an agent to enhance the present solution while trading between exploitation and exploration of the solution search space. Exploration is a local search that improves on an existing solution by concentrating on the few promising regions of the solution search space. That is when an agent acts with exploitation, the action with the best Q-value will be greedily selected (greedy strategies). One of the main disadvantages of exploitation is that the search can be stuck about in a local optimum because the action selection may never explore actions with low Q-values. Conversely, exploration is a global search that improves an existing solution by considering a larger size of the search space to explore new promising solutions that are yet to be considered.

This implies that if an agent chooses an action with exploration, a random action will be selected to gain new knowledge concerning the environment and possibly find superior actions. Therefore, exploration is more likely to result in a global optimum. Though an agent chooses the best rewarding action with exploitation at a one-time step, the total rewards are maximized with exploration in the long run. Therefore, for the convergence of the Q-value that will lead to an optimal policy, a trade-off is required between the exploitation and exploration when RL is applied to a large solution space. In achieving this trade-off, most heuristics that scales well in term of cost are mostly used which are as follows[76, 140].

(i) Greedy strategy: This technique is solely exploitation where the action with the highest Q-value is selected. That is:

$$A_t = \underset{a \in A}{\text{argmax}}\{Q_t(s,a)\} \tag{2.21}$$

It is important to note that the greedy strategy alone may not converge to the global optimum. This is because actions with initial low Q-values will not be explored.

(ii) ϵ - greedy strategy: This strategy is a randomized strategy that selects the action with the highest Q-value using Equation 2.21, with the probability $\epsilon \in [0,1]$ that a random action is selected from the eligible actions from the current state in order to explore new actions. Nevertheless, this technique may lower the exploited reward after a long period of learning.

(iii) Probability distribution based strategy: This is also known as the Boltzmann exploration that selects an action based on converting the Q-values into selection probabilities for every action of every state and samples are carried out on the results using the Boltzmann probability distribution given as:

$$Pr\{A_t = a\} = \frac{e^{\frac{Q(s,a)}{\tau}}}{\sum_{b \in A} e^{\frac{Q(s,b)}{\tau}}} \tag{2.22}$$

τ is known as the temperature parameter. A low value of τ favors exploitation while a high value of τ favors exploration. This method is clearly suitable when the best action is clearly distinguished from others. However, it suffers degradation in its performance when the actions close values.

(iv) Interval-based strategies: This method controls exploration by storing the statistics of the number of successes and trials of each action. An action is selected based on its Q-value and the potential of the action success probability using the:

$$A_t = \underset{a \in A}{\text{argmax}}(Q_t(s,a) + cb^+) \tag{2.23}$$

The b^+ is a bonus that decreases over time and is obtained by counting the number of times a state has been visited $N(s)$ and the selection of the corresponding state-action combination $N(s,a)$. More specifically b^+ is given as:

$$b^+ = \sqrt{\frac{\ln N(s)}{N(s,a)}} \tag{2.24}$$

$c > 0$ is a parameter used to adjust the exploration degree. The higher the value of c, the more the bonus b^+ has more leverage thereby resulting in an increase in the explorative character.

2.3.4 *Q-Routing Protocol*

RL was first applied to solving routing problems in networks by Boyan and Littman in 1994 and the name of the routing protocol was called Q-Routing [25]. Q-routing protocol is an adaptive protocol for routing packets in networks. The protocol achieves a balance between load balancing to avoid network congestion and the baseline shortest-path

routing along popular routing paths in the network by trying different routing policies and gathering statistics to compare which routing decisions yield better delivery times.

Q-Routing is a distributed protocol because each node only uses local communication with its direct neighbors to evaluate its actions. Q-Routing protocol can cope with changing network conditions like dynamic network loads and dynamic topologies because it is online and adaptive.

Q-routing protocol uses Q-learning to minimize the delivery delay of packets from the sources to their destination. Let $Q_i(d, j)$ represent the delivery delay estimated for node i to send a packet to destination node d through its neighbor node j. The delivery time includes all delays that the packet experiences which includes the processing time, queue time, and transmission time.

This implies that a node, i generating or/holding a data packet to send to the sink, d through the neighbor, j estimates the delivery delay, $Q_i(d, j)$ which it takes for the next forwarder to route the packet to the sink. Since every node maintains a neighboring table that contains its Q-value to the destination, when a sensor node, i has a packet to send to the sink, it chooses the next forwarder, k with the minimum Q-value using greedy strategy. The minimum Q-value is estimated as:

$$\tau_j = \min_{k \in \mathcal{N}_j} \{Q_j(d, k)\} \tag{2.25}$$

Then node i updates its Q-value which is the estimated delivery time associated with the neighbor node j using:

$$Q_i^{new}(d, j) \leftarrow (1 - \alpha) Q_i^{old}(d, j) + \alpha \left[R_{i,j} + \tau_j \right] \tag{2.26}$$

where $R_{i,j}$ is the immediate reward require to send packets from node i to node j, α is the learning rate, and the discount factor γ is 1.

The immediate reward $R_{i,j}$ is given as:

$$R_{i,j} = \vartheta_i + \chi_{i,j} \tag{2.27}$$

where ϑ_i is the queuing time of the packets at node i and $\chi_{i,j}$ is the link cost (transmision time) between node i and node j.

The pseudocode of the Q-Routing protocol is given in the Algorithm 1

Algorithm 1 Q-Routing Protocol

1: Initialize the Q-value matrix of node i, $Q_i(*, *)$.
2: **while** Until the terminal condition is reached **do**
3: **if** Node i has packet to send **then**
4: Select neighbor node j with lowest Q-value.
5: Send packet to next forwarder node j
6: Feedback is received by node i from node j with Equation 2.27.
7: Node i updates it Q-value with Equation 2.26.
8: **end if**
9: **end while**

However, Q-routing suffers from slow convergence, Q-value freshness (a node has no accurate path quality when the route is not used for a long time in routing, which leads to the non-optimal selection of the next forwarder), and it is very sensitive to the parameter setting.

2.4 GENETIC ALGORITHM

A Genetic Algorithm (GA) is a search-based heuristic technique based on the concept of natural selection and genetics. GA is a subset of evolutionary algorithms proposed by Charles Darwin. Evolutionary algorithms are inspired by the theory of biological and natural evolution. GA was developed by John Holland and David E. Goldberg at Michigan University. GA is used to find optimal or sub-optimal solutions to NP-hard optimization problems [86].

GA imitates the procedure of natural selection. This means that individuals who can adapt to changes in their environment can survive, reproduce and go to the next generation in line with the Darwinian Theory of "Survival of the Fittest". Five phases are considered in GA. These are initial population, fitness function, selection, crossover, and mutation. The GA terminates if it convergences or the number of generations specified is reached [82].

2.4.1 *Initial Population*

GA starts with a set of possible solutions to the given optimization problem called population [90]. A solution is an individual that is defined by a set of variables called Genes. The combination of genes into a string results in the formation of a chromosome (solution). The set of genes of a chromosome is denoted using a string. Mostly, binary digits are used (a string of 0s and 1s). Therefore the genes are encoded in a chromosome. Figure 2.7 gives the relationship between a population, chromosomes, and genes.

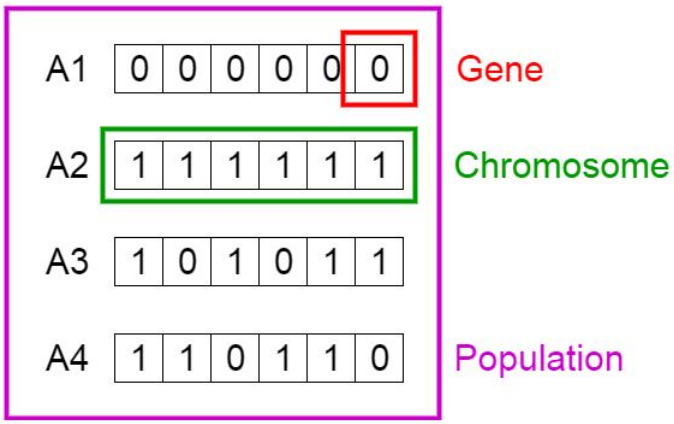

Figure 2.7: Population, Chromosomes, and Genes.

2.4.2 *Fitness Function*

The fitness function is a function that takes an individual from a population as a candidate solution to a given optimization problem and evaluates how "fit" the solution is with reference to the optimization problem. The fitness function gives the probability of

an individual being selected as a parent for reproduction for the next generation. The probability value is known as the fitness score. Therefore the individuals with the optimal fitness score are selected for mating in the next generation [69].

2.4.3 *Selection*

This is a phase in the genetic algorithm that involves choosing the two fittest parents from the population to breed for the next generation. The most popular technique of parent selection is known as fitness proportionate selection [10]. In fitness proportionate selection, the probability of selecting an individual for breeding is proportional to its fitness score. That is the probability, p_i of selecting the i^{th} individual is given as in Equation 2.28 [94].

$$p_i = \frac{f_i}{\sum_{j=1}^{N} f_j} \tag{2.28}$$

where N is the size of the current population.

In the case where the individuals of the population have the same fitness score, the parents can be selected randomly. This method of selection is known as random selection.

2.4.4 *Crossover*

Crossover is a phase of a genetic algorithm that is used to combine the genes of two parents from an existing population to form a new child. Crossover is done by randomly choosing a crossover point among the genes of the parents and subsequently interchanging the genes until the crossover point is reached [90]. This leads to the formation of a new child that is added to the population. The crossover operation is shown in Figure 2.8.

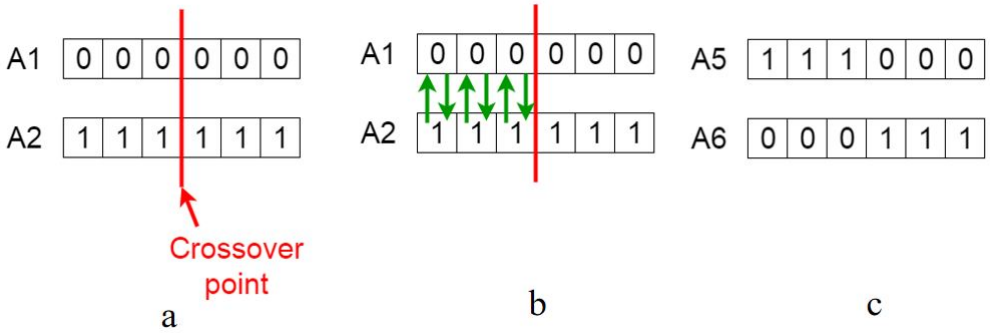

Figure 2.8: Crossover Operations

2.4.5 *Mutation*

The mutation is a genetic operator that results in a new solution by randomly flipping the genes in a chromosome with a probability greater than a random variable [10]. The mutation prevents premature convergence by introducing diversity to the population. The mutation operation is shown in Figure 2.9.

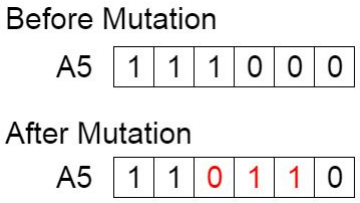

Figure 2.9: Mutation Operations

The standard GA algorithm is presented in the Algorithm 2 [126].

Algorithm 2 Genetic Algorithm

1: Define fitness function, FF
2: Initialize the number of generation, $t = 0$
3: Create random individuals in the initial population, $P(t)$
4: Evaluate individual fitness in $P(t)$ using FF
5: **while** Termination condition is not reached **do**
6: $t = t + 1$
7: Select individuals from population $P(t-1)$ to $P(t)$
8: Generate new individuals from $P(t)$ with crossover and mutation
9: Evaluate individual fitness in $P(t)$ with FF
10: **end while**
11: **return** best individuals found during the evolution.

2.5 GRAPH THEORY

Graph theory entails the study of graphs. A graph is a mathematical structure that models pairwise relations between objects. A graph in the context of WSNs consists of vertices which are the sensor nodes and edges which are the wireless links that connect two unique sensor nodes. A graph can be classified as an undirected graph or a directed graph. For an undirected graph, The edge connection between two vertices is symmetrical, while for a direct graph, the edge connection between two vertices is asymmetric.

A graph can be represented mathematically as $G = (V, E)$, where V is the set of vertices and $E \subseteq \{\{u, v\} \mid u, v \in V \text{ and } u \neq v\}$ is the set of edges, which are unordered pairs of vertices (this means that an edge is associated with two distinct vertices.

An undirected graph of an unordered pair of vertices $\{u, v\}$ is called connected if a path leads from u to v. Otherwise, the unordered pair is called disconnected. That is, a connected graph is an undirected graph in which every unordered pair of vertices in the graph is connected. Otherwise, it is called a disconnected graph.

2.5.1 *Minimum Spanning Tree*

A tree is an undirected graph in which any two vertices are connected by exactly one path. In other words, a tree is a graph without cycles and each edge is a bridge. A spanning tree is a tree that connects all the nodes of a graph without forming a cycle. A minimum spanning tree of a graph is a spanning tree that has a minimum total edge weight. A minimum spanning tree may not be unique in a graph with an integer edge. Therefore, a graph can have more than one minimum spanning tree with the same minimum total edge weights.

The classical algorithms for finding MST of a connected undirected graph are Kruskal's algorithm [87], Prim's algorithm [113], and Boruka's algorithm [44]. The classical MST algorithm is explained in the sequel.

2.5.1.1 *Kruskal's algorithm*

Kruskal's algorithm belongs to the class of greedy algorithms. Kruskal's algorithm constructs an MST considering only the edge having minimum weight among all available edges. Given a weighted nontrivial graph $G = (V, E)$ that is connected, let $w : E \rightarrow R$ be the weight function of G. The first stage is creating a "skeleton" of the tree T that is initially set to be a graph without edges, that is $T = (V, \phi)$.

The next phase entails sorting the edges of G by weights in nondecreasing order. In other words, the edges of G are labeled as $E = \{e_1, e_2, ..., e_n\}$. Where $w(e_1) \leq w(e_2) \leq \ldots \leq w(e_n)$ and $n =\mid E \mid$. For each edge, e_i, Kruskal's algorithm adds e_i to T if e_i does not result in T having a circle. Kruskal's algorithm runs in $O(\mid E \mid log \mid E \mid)$ time. The pseudocode of Kruskal's algorithm is given in Algorithm 3.

Algorithm 3 : Kruskal's algorithm

Input: A connected graph $G(V, E)$ having w as the weight function.
Output: An MST of G.

1: $m \leftarrow \mid V \mid$
2: $T \leftarrow \phi$
3: sort $E = \{e_1, e_2, ..., e_n\}$ so that $w(e_1) \leq w(e_2) \leq \ldots \leq w(e_n)$
4: **for** $i \leftarrow 1, 2, \ldots, n$ **do**
5: **if** $e_i \notin E(T)$ and $T \cup \{e_i\}$ is acyclic **then**
6: $T \leftarrow T \cup \{e_i\}$
7: **if** $\mid T \mid = m - 1$ **then**
8: Return T
9: **end if**
10: **end if**
11: **end for**

2.5.1.2 *Prim's algorithm*

Prim's algorithm, like Kruskal's algorithm, adopts a greedy method to compute an MST of a connected weighted graph $G = (V, E)$, where $n = |V|$ and $m = |E|$. Prim's algorithm runs in $O(n^2)$ time. The pseudocode of Prims's algorithm is given in Algorithm 4. For individual $v \in V$, let cost$[v]$ be the minimum edge weight of all edges linking v to a vertex in the tree T, and parent$[v]$ be the parent of v in T.

Prim's algorithm organizes vertice v not contained in T in the minimum-priority queue Q, according to prioritized cost$[v]$. $cost[v]$ is first set to a number that is larger than any weight in the graph G, which is usually infinity and the parent of every vertice is set to NULL since the construction of the MST T has not started as provided in Lines 1 to 3. From lines 4 to 6, an arbitrary vertex r is chosen from V which is the root of T, and Q is set to be all vertices from V.

At this stage cost$[r]$ is set as zero, this makes r to be the only vertice with a cost that is less than infinity. In lines 7 to 12, during the first execution of the while loop, r is the first vertice extracted from Q to process. Line 8 extracts a vertex u from Q based on the key cost, thus moving u to the vertex set of T. Line 9 takes all vertices adjacent (neighbors) to

u into consideration. The while loop updates the cost and parent fields of every vertex v adjacent to u that is not in T. If parent$[v]$ is not equal to NULL, then cost$[v]$ is less than infinity, and cost$[v]$ is the weight of an edge linking v to a vertex already in T. Lines 13 to 14 generate the edge set of the MST and return this edge set.

Algorithm 4 : Prim's algorithm

Input: A connected graph G(V, E) having w as the weight function.
Output: An MST, T of G.

```
1:  for each v ∈ V do
2:     cost[v] ← ∞
3:     parent[v] ← NULL
4:  end for
5:  r ← any vertex of V
6:  cost[v] = 0
7:  Q ← V
8:  while Q ≠ ϕ do
9:     u ← extractMin(Q)
10:    for each v ∈ adj(u) do
11:       if v ∈ Q and w(u, v) < cost[v] then
12:          parent[v] ← u
13:          cost[v] ← w(u, v)
14:       end if
15:    end for
16: end while
17: T ← {(v, parent[v] | v ∈ V − {r})}
18: return T
```

2.5.1.3 *Boruvka's algorithm*

Boruvka's algorithm is an algorithm that builds an MST in a connected weighted graph $G = (V, E)$ that has distinct edge weights. Boruvka's algorithm runs in $O(m\ log\ n)$ time, where $n =\mid V \mid$ and $m =\mid E \mid$. The pseudocode of Boruvka's algorithm is given in Algorithm 5. A spanning forest T of G is initialized in lines 1 and 2. The spanning forest T is a subgraph of G that contains all the vertices and no edges of G.

The trivial graph K_n represents the n components of the initial forest. A spanning tree of G is constructed through a recursive procedure with the while loop contained in lines 3 to 6. For every component, $T^{'}$ of T, the algorithm considers every out-going edge of $T^{'}$ and selects an edge $e^{'}$ with minimum weight among all such edges and is added to the edge set of T. This makes two different trees to be joined together by a bridge. The final graph at the end of the loop will be an MST of G.

Algorithm 5 : Boruvka's algorithm

Input: A connected distinct weighted graph $G(V, E)$ having w as the weight function.
Output: An MST of G.

```
n ←| V |
T ← K_n
while | E(T) | < n − 1 do
    for each component T' of T do
        e' ← minimum edge weight that leaves T'
        E(T) ← E(T) ∪ e'
    end for
end while
return T
```

2.6 REVIEW OF SIMILAR WORKS

The first hop-by-hop routing protocol to utilize RL is called Q-routing proposed by Boyan and Littman, (1994) [25]. Q-routing minimizes the packet delivery delay. However, Q-routing suffers from Q-value freshness and slow convergence, and it is very sensitive to parameter setting. Different routing protocols such as [28, 29, 49, 88, 111, 130] have also applied RL to optimize network delivery delay.

Different authors have done network lifetime and energy optimization routing protocols based on RL for WSNs. The sequel presents some of these works.

Zhang and Fromherz, (2006) designed RL-based constrained flooding for WSNs to optimize the number of packets transmitted when sending data packets from source nodes to the sink [148]. The cost of flooding is reduced by using Q-learning to learn the packet sending cost which can be a delivery delay, hop counts, etc., thus enabling energy saving. The estimated cost of the sender which is captured in the Q-value is encapsulated in each data packet. The action of the RL-based constrained flooding is packet broadcasting using either constrained propagation, differential delay, or probabilistic retransmission without using control packets. Simulation results showed that the RL-based constrained flooding has improved energy efficiency when compared with direct routing [99] and backbone tree [62] that uses direct diffusion [70]. However, direct routing has better packet delivery delay than RL-based constrained flooding.

Zhang and Huang, (2006) designed an RL-based and constraint-based routing strategy for WSNs called an adaptive tree routing protocol [149]. The adaptive tree routing protocol maximizes the network lifetime taking into consideration, load balancing, network congestion mobile sinks, and link failures using Q-learning. The parameters used for finding the optimal route for a given application include the maximum number of failed confirmations retransmissions, Q-values learning rate, NQ-values updates rates, and reset threshold for a parent. Simulation results show that adaptive tree routing protocol can find the best routing tree to the sink despite sink mobility and node failures without incurring more control packets for tree maintenance. The adaptive tree routing protocol has improved lifetime, latency, and delivery ratio when compared with backbone tree [62] and grid routing [30] protocols.

Wang and Wang, (2006) proposed a routing algorithm called Adaptive Routing for WSNs using RL (AdaR) to maximize the network lifetime [137]. The protocol uses the multiple factors of hop count, residual energy, link reliability, and the number of routing paths crossing a node to determine the optimal routing path. AdaR converges faster

than Q-routing to the optimal solution and does not suffer from the problem of initial parameter setting. However, there was no explicit definition of the network lifetime. Also, AdaR has a high computation complexity when compared with Q-routing. This is because AdaR required the collection of samples to be used for learning.

Zhang and Fromherz, (2006) proposed an energy-efficient routing protocol based on RL for WSNs called constrained flooding [148]. The protocol enables energy saving by adapting Q-routing to optimize the cost of sending data packets to the sink in WSNs with flooding. This reduces the number of packet transmissions and a corresponding reduction in the energy consumption of the WSNs.

Nurmi, (2007) proposed an energy-aware and selfishness RL-based routing protocol for ad hoc networks [107]. The protocol uses RL, function approximation, and stochastic approximation to choose the next forwarder. The protocol provides a generic model to evaluate the node energy consumption, ratio of packet re-forwarding, and selfishness. This enables the dynamic association of a forwarding probability to each of the nodes' neighbors. However, the selfishness and energy function was not provided, since the protocol is generic.

Dong et al., (2007) proposed for ultra-wideband sensor networks a Reinforcement Learning Based Geographical Routing Protocol (RLGR) [41]. The protocol seeks to improve the network lifetime by reducing packet delivery delay and distributing energy consumption among nodes uniformly. RLGR considers hop counts to the sink and residual energy of nodes in choosing the next forwarder. RLGR improved the network lifetime by at least 75 percent when compared with Greedy Perimeter Stateless Routing (GPSR)[78] by simulation. However, RLGR suffers from slow convergence to the optimal routing paths due to its distributive nature.

Arroyo-Valles et al., (2007) proposed for WSN a geographical routing algorithm called Q-Probabilistic routing (Q-PR) [18]. Q-PR uses RL and a Bayesian decision model to make routing decisions based on a delayed reward of previous actions and the immediate interaction between neighboring nodes. Q-PR maintains the trade-off between network lifetime and the expected number of re-transmissions while increasing the packet delivery ratio. But, Q-PR suffers the following limitations: the message's importance is not balanced with the energy cost of using a constant discount factor of one, the selection of the next forwarder requires the requisites of neighbors and non-refinement of the estimation of the residual energy of the sensor nodes.

GhasemAghaei et al., (2007) designed two adaptive swarm intelligence-based routing algorithms for WSNs which are Adaptive Routing (AR) algorithm and the Improved Adaptive Routing (IAR) algorithm [50]. AR algorithm is a modification of the adaptive Ant-based Dynamic Routing (ADR) for packet-switched communication networks. The modification was achieved by replacing the RL queue parameters of the ADR algorithm [95] with the WSN RL parameters. However, the AR algorithm did not converge to an optimum solution. The IAR algorithm improved the AR algorithm by adding the cost between the neighbor node and the sink and a coefficient. The performance of the AR algorithm and the IAR algorithm is validated by comparison with Flooded Piggybacked Ant Routing, Sensor-driven Cost-aware Ant Routing [150], and Basic Ant Routing [36], in terms of latency, success rate, energy consumption, and energy efficiency. AR yields good performance results, and the IAR yield the best performance results in every tested condition.

Naruephiphat and Usaha, (2008) proposed for Mobile ad hoc network (MAGNET) a routing protocol for balancing the tradeoff between minimizing energy consumption and maximizing network lifetime [105]. This is achieved by using RL to select routing paths

based on the energy consumption of paths and the residual energy of nodes. The protocol yields a high ratio of packet delivery using low network energy consumption and thereby promotes in the long run, network lifetime. The network lifetime considered is the time when the first node depletes its energy source, however, sensing is still possible unless the node with the depleted energy source is the sink.

Förster and Murphy, (2008) designed a distributed multicast routing algorithm based on RL called E-FROMS [48]. E-FROMS balanced the energy consumption in a multiple sinks WSN by learning the optimal spanning tree that minimizes the energy-based reward. The reward is a function of the hop counts of a path and the minimum sensor nodes' residual energy on the path for sending packets from a source node to multiple mobile sinks. Each sensor node is an agent with its state as a set of sinks, and for each sink, the set of paths to that sink. The action is to select the next hop to send packets to multiple destinations using the epsilon-greedy exploration policy. The strength of E-FROMS is that the communication overhead is low which enables the achievement of good bandwidth utilization. However, the state space and action space overhead is high and very high, respectively.

Wen et al., (2008) proposed an Energy and Delay model using ant algorithms (ED ANTS) to minimize the delivery delay in an energy-efficient manner in a round of data transmission for WSNs [141]. RL-algorithm is used to train the energy and delay model in order to provide a trade-off between the network energy consumption and delay of the WSNs. This caused the maximization of the network lifetime while minimizing the delivery time of the data transmission. Simulation using the OPNET simulator shows that ED ANTS performs better than existing ant-based routing protocols which are AntNet [27] and AntChain [38] in terms of delay and energy cost per round.

Hu and Fei, (2010) proposed for underwater sensor network (UWSN) a Q-learning-based energy-efficient and lifetime-aware routing (QELAR) for finding the optimal routing path in the network [64]. The protocol makes the residual energy of the nodes to be distributed evenly and thereby increasing the network lifetime. The packets in the network are forwarded based on a reward function that takes into consideration the energy distribution of a group of nodes and the residual energy of each node. The limitations of QELAR are high overhead due to control packets and slow convergence to the optimal routing paths due to the protocol's distributive nature.

Sharma et al., (2012) proposed a tailored Q-learning-based approach for energy-efficient routing in WSNs [124]. The main aim of the protocol is to use an improved Q-Learning algorithm to minimize the energy consumed by wireless sensors. The protocol enables sensor nodes to learn the best routing strategy to efficiently route when performing data aggregation to the sink.

Yang et al., (2013) proposed a reinforcement learning-based routing protocol between sensor nodes and mobile sinks, which are vehicles [145]. The protocol enables the direct interaction between the sensor nodes and the mobile sinks taking multiple metrics such as residual energy, and hop count in learning the routing paths. However, the protocol incurred high overhead due to control packets.

Oddi et al., (2014) extend the Q-Routing protocol, designed for wired networks, to be utilized in WSNs [108]. The proposed routing protocol called optimized Q-Routing optimizes the network lifetime, by balancing the routing load among the sensor nodes, taking into consideration the sensor nodes' current residual energies while minimizing the control overhead. However, OPT-EQ-Routing requires too many iterations to converge to the optimal paths.

Jafarzadeh and Moghaddam, (2014) proposed a routing protocol for WSNs called Energy-aware QoS routing RL-based (EQR-RL) [73]. EQR-RL optimizes the energy in WSNs while guaranteeing the delivery delay of packets. EQR-RL employs the probability distribution-based exploration strategy to choose the next hop to forward data packets. The reward function of the protocol is based on weighted metrics of selected forwarder residual energy, link delay, and the ratio of packets between the packet sender and the selected forwarder. But, EQR-RL suffered from high convergence time to the optimal route.

Guo et al., (2014) proposed an intelligent routing protocol for WSNs built on RL named reinforcement-learning-based lifetime optimization (RLLO) routing protocol [54]. RLLO uses the residual energy of the sensor node and hops count to the sink in its reward function to update agents' Q-values. The agents in RLLO are the sensor nodes. The routing protocol is implemented in NS2. Simulation results show improved performance when compared with energy-aware routing (EAR) and improved energy-aware routing (I-EAR) using network lifetime and packet delivery as performance metrics. However, RLLO is prone to a very high probability of network isolation.

Patel and Shah, (2015) designed a shortest-path Q-routing protocol for WSNs to minimize energy consumption and maximize the network lifetime [110]. The protocol learns the shortest link distance route from the source sensor nodes to the sink using Q-learning. The shortest link distance route concept is taken from Dijkstra's algorithm [37] used for routing data packets in networks. The shortest-path Q-routing protocol network lifetime is compared with existing Energy-aware Q-routing protocol [35] on homogeneous topologies, heterogeneous topologies, and hybrid topologies. The simulation results show that the shortest-path Q-routing protocol performs better than the Energy-aware Q-routing protocol for all topologies. However, the shortest path Q-routing protocol can be improved to increase the network lifetime by integrating a Q-Routing Compression (Q-RC) protocol [21] that learns the best path to aggregate and compresses data packets from the sources before routing toward the sink.

Debowski et al., (2016) proposed a hybrid protocol called Q-Smart Gradient-based routing protocol (QSGrd) for WSNs [34]. QSGrd optimizes the energy consumption in WSNs by combining transmission gradient and Q-learning. In QSGrd, each neighbor of a node is associated with transmission success probability which depends on the maximum transmission range and the distance between nodes. The transmission success probabilities of the neighbors of a node result in a transmission gradient. Subsequently, the transmission probabilities are used to update the Q-values. The optimal routing paths are learned using the average least number of transmissions to the sink and the residual energy of the next hop with RL. However, QSGrd suffered these limitations: slow convergence to the optimal routing paths, the static parameter of the Q-learning leads to network performance degradation, and increased computation time.

Le and Sangman, (2017) designed an RL-based Communication Range Control (RL-CRC) scheme for WSNs [92]. The scheme uses Q-learning to maintain the optimal degree of each sensor node with the aim of adapting the communication range of each sensor node while maintaining the connectivity of the network for the dynamically changing WSNs. This enabled the minimization of interference, transmission power, and energy consumption at the sensor nodes. Simulation results showed that RL-CRC has improved reduction in energy consumption when compared to classic communication range control schemes which are Fuzzy-logic Topology Control (FTC) [66] and Local Tree-based Reliable Topology (LTRT) [101] while maintaining nearly equal average communication range.

Renold and Chandrakala, (2017) proposed for WSNs a routing protocol called Multi-agent Reinforcement Learning-based Self-Configuration and Self-Optimization (MRL-SCSO) [119]. In this protocol, the reward function is defined using the buffer length and the node residual energy. The next forwarder selected is the neighbor with the maximum reward value. The protocol also incorporates the sleeping scheduling scheme to decrease the energy consumption of nodes. The network lifetime of MRL-SCSO is higher than that of the Collect Tree Protocol (CPT) [51] when compared by simulation. But, MRL-SCO requires an increased number of episodes to learn the network.

Akter and Yoon, (2018) proposed an RL-based protocol for duty cycle interval control in WSN [5]. The protocol used Q-learning to find the maximum duty cycle interval that supports various delay requirements while maintaining the given delay success ratio. The agent of the protocol resided in the sink of the WSN. Simulation results showed that the protocol reduces the energy consumption of sensor nodes and maximizes the network lifetime while guaranteeing the required delay and delay success ratio.

Soni and Shrivastav, (2018) designed two algorithms which are the RL-based clustering algorithm (RLBCA) and the on-demand mobile sink traversal (ODMST) algorithm to maximize the network lifetime while minimizing the network energy consumption [129]. The RLBCA creates optimal cluster heads using Q-learning. The learning agent of the RLBCA is the sensor nodes that use the nearest neighbor energy level to form clusters. This enables the sensor nodes to learn the cost of the route to send data packets to the cluster head. The ODMST algorithm enables the mobile sink to collect data from the cluster heads based on demand to improve the network lifetime. Simulation results showed that RLBCA and ODMST algorithm improves the network energy consumption and lifetime when compared with Two-Tier Data Dissemination (TTDD) [96], Delay Bound Reduced k-Means (DBRkM) [81], Energy-efficient Particle swarm optimization based routing algorithm with Mobile Sink (EPMS) [136] RLLO and RL-CRC.

Kim et al., (2019) proposed an RL-based topology-aware routing protocol for underwater WSNs [84]. The protocol takes into account the local topology of sensor nodes to enable the sensor node to transmit information in the right direction. The protocol also considers the changes in channel status and the residual energy of the sensor node so as to minimize the network energy consumption. The simulation result shows that the protocol performs better than QELAR in terms of total energy consumption and latency.

Geo et al., (2019) proposed for WSN a Q-learning routing protocol called a Reinforcement Learning-Based Routing (RLBR) to optimize the network lifetime [55]. RLBR search for optimal paths for transmitting packets from each node to the sink taking into consideration of hop count, link distance, and residual energy in its reward function. RLBR utilizes transmit power adjusting and data packet carrying feedback scheme to increase packet delivery, balance the energy consumption, and reduce the overall energy consumption. RLBR performs better than Q-Routing, and MRL-SCSO in terms of network lifetime and energy efficiency. However, RLBR suffers from slow convergence to the optimal routing paths.

Bouzid et al., (2020) proposed a routing protocol for WSNs known to optimize lifetime and energy consumption [24]. R2LTO learns the optimal paths to the sink by considering the hop count, residual energy, and transmission energy (distance) between nodes. R2LTO consists of two processes, which are the discovery process to know the network topology and the continuous learning routing process. The effectiveness of R2LTO is carried out by comparison with Q-routing and RLBR by simulation, and the results show that R2LTO performs better in terms of network lifetime and energy efficiency. However, R2LTO suffers from slow convergence to the optimal routing paths.

Kaur and Aulakh, (2021) designed an RL-based LEACH protocol (RL-LEACH) for WSNs [83]. An optimal cluster number of the network is first evaluated by analyzing the energy consumption of the network for both inter and intra-clustering communication. The protocol uses Q-learning to enable each sensor node to select an optimal Cluster Head (CH) by taking into consideration the distance and energy consumption required to send data from the sensor node to the CH. Simulation results showed that RL-LEACH has improved performance in energy consumption, network lifetime, and packet delivery ratio when compared to the classical LEACH protocol.

Sapkota and Sharma, (2021) designed an RL-based routing protocol to optimize the network lifetime in WSN [121]. The agents which are the sensor nodes choose the next forwarder with Q-learning by using the inverse of the distance between connected sensor nodes as the reward function. Simulation results showed that the proposed protocol has improved performance when compared with the baseline direct diffusion protocol [71]. But, the protocol requires an increased number of episodes to learn the network.

Mutombo et al., (2021) proposed an RL-based Energy Balancing Routing (EBR-RL) protocol for WSNs [103]. EBR-RL protocol maximizes the network lifetime by balancing the energy consumption between sensor nodes. EBR-RL protocol operates in two stages. The first stage set up the network and the second stage carries out the data transmission using RL. EBR-RL protocol has better performance in terms of network lifetime and energy saving when compared with existing energy-efficient routing protocols. However, EBR-RL suffers from slow convergence to the optimal routing paths.

Abadi et al., (2022) designed an RL-Based Energy Efficient Control and Routing Protocol (RLBEEP) for energy management in WSNs [2]. RLBEEP improves the lifetime of the network using three energy management techniques. The first technique uses Q-learning to learn the route with the minimum length that reduces the energy consumption of the sensor node. The second technique uses a sleep schedule to improve the energy consumption of sensor nodes. The last technique is the adaptation of sensor node data transmission based on the received data rate change. Simulation results show that RLBEEP performs better than RLBR and Delay-aware data fusion (DADF) [42] protocols in terms of network lifespan and throughput.

Joshi and Kumar, (2022) designed an RL-based energy-aware routing protocol to maximize the lifetime and throughput of multimedia WSN [75]. The protocol first utilizes adaptive clustering to select cluster heads considering the distance and average remaining residual energy of heterogeneous sensor nodes. After the adaptive clustering, the protocol learns the route for inter-cluster data aggregation so that the network lifetime is maximized while minimizing the network energy consumption using a State-Action-Reward-State-Action (SARSA). Simulation results show that the designed protocol has improved network lifetime and throughputs when compared with existing routing protocols such as Sleep-awake energy-efficient distributed (SEED) clustering [4], Firefly-based hierarchical maximum likelihood (FHML) [52], Harmony search-based particle swarm optimization (HPSO) [123], and GA-based energy-efficient clustering (GEEC) [138].

Uddin, (2022) proposed an RL-based data aggregation protocol for CHs (RL-CH) in wireless multimedia sensor networks (WMSN) to maximize energy efficiency and network lifetime [134]. RL-CH protocol selects CHs using evolutionary game theory and learns the best path between the CHs to send data to the sink using Q-learning considering hop count and residual energy of the CHs in the reward function. The state of the agent of the RL-CH includes all CHs and the sink. The action of the agent is to select the best neighboring CH to forward data packets toward the sink. Simulation results show that the RL-CH protocol performs better in terms of end-to-end delay,

packet delivery ratio, energy efficiency, and network lifetime when compared with the evolutionary game-based routing (EGR) protocol [56] serving as a baseline.

3

ANALYSIS OF IMPACT OF Q-LEARNING PARAMETERS ON DISTRIBUTED RL-BASED LIFETIME-AWARE ROUTING PROTOCOLS PERFORMANCE

3.1 INTRODUCTION

RL has been used in WSNs to balance energy consumption, reduce energy consumption and maximize the network lifetime. This is because RL can find the best routing path in WSNs considering the residual energy of sensor nodes, link distance of sensor nodes to the sink, and hop counts of sensor nodes to the sink. This chapter entails the analysis of the impact of Q-learning parameters which are learning rate and discount factor on selected distributed RL-based lifetime-aware routing protocols performance for WSNs on the environment designed with Networkx.

3.2 METHODOLOGY

This section presents the network model, routing and learning process, and the energy model used for the performance analysis of the distributed lifetime-aware routing protocols.

3.2.1 *Network Initialization Procedure*

The WSN consists of a set of sensor nodes and a sink. After the network initialization, each sensor node broadcasts its status information which includes a unique identifier (ID), $x - y$ location, load (number of data octets to transmit per second to the sink), residual energy, and maximum transmission range. The sink collects all the sensor nodes' status information and builds the network graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{v_1, ..., v_n\}$ is the set of sensor nodes and $\mathcal{E} = \{e_1, ..., e_m\} \subseteq \mathcal{V} \times \mathcal{V}$ is the set of network links between two distinct connected nodes in the network.

Two sensor nodes are only connected if their cartesian distance is less than or equal to the maximum transmission range of the sensor nodes. The initial path quality of each sensor node is calculated as the ratio of the initial residual energy and the hop count to the sink. The sink computes the minimum hop count from each sensor node by constructing a minimum spanning tree graph using Prim's algorithm with a unit distance between two connected nodes. This minimum hop count is sent to each sensor node by the sink using a notification packet. Each sensor node acts as an agent and initializes the Q-value which gives the path quality using Equation 3.1. Each sensor node, s_v encodes the Q-value on the notification packet and broadcasts it to all neighboring sensor nodes.

$$Q(s_v) = \frac{E_R(s_v)}{H(s_v)} \tag{3.1}$$

where $Q(s_v)$ is the Q-value, $E_R(s_v)$ is the initial residual energy, and $H(s_v)$ is the hop count to the sink of the sensor node, s_v.

This enables each sensor node to initialize its neighbor information which includes the sensor node's ID, $x-y$ location, residual energy, hop count, and Q-value on its neighboring table.

3.2.2 *Routing and Learning Procedure*

After the network initialization process, the sensor nodes not only generate data packets for transmission but also act as a router. Each sensor node is an agent that learns the best routing path to the sink. The state space of an agent is the set of neighboring sensor nodes with the state as the sensor node holding the generated packet. Consequently, the action space of an agent is the set of neighboring sensor nodes' IDs, and the action is to select a neighboring sensor node as the next forwarder.

A sensor node, s_v generating or/holding a data packet, P to send to the sink, s_d through the next forwarder, s_f estimates the path quality, $Q_{s_v}(s_d, s_f)$ which it takes for the next forwarder to route the packet, P to the sink. Since every sensor nodes maintain a neighboring table that contains its Q-value and hop count to the sink, when a sensor node, s_v has a packet to send to the sink, it chooses the next forwarder, s_f with the maximum Q-value using the epsilon greedy strategy. This is to ensure the exploration of the solution search space. That is given a probability value of epsilon, $\epsilon \in [0,1]$ and a random number, $r \in (0,1)$ generated in each learning round, the next forwarder, s_f is selected using Equation 3.2.

$$s_f = \begin{cases} \textit{Random action, } if\ r \geq 1-\epsilon \\ \underset{s_f \in Ng(s_v)}{\text{argmax}}\{Q_{s_v}(s_d, s_f)\}, \textit{ otherwise.} \end{cases} \tag{3.2}$$

This is because when the packet, P is sent to s_f by s_v, the sensor node, s_v receives a reward which is used to measure the feedback from s_f that gives the path quality to get the sink from s_f using Equation 3.3.

$$Q_{s_f}(s_d) = \max_{s_k \in Ng(s_f)} \{Q_{s_f}(s_d, s_k)\} \tag{3.3}$$

where $Ng(s_f)$ is the set of the next forwarder neighboring sensor nodes. Subsequently, the sensor nodes, s_v update their path quality and hop count using Equations 3.4 and 3.5, respectively.

$$Q(s_v) = Q_{s_v}(s_v, s_f) \tag{3.4}$$

$$h(s_v) = h(s_f) + 1 \tag{3.5}$$

The path quality, $Q_{s_v}(s_d, s_f)$ which it takes for the next forwarder to route the packet, P to the sink is updated using Q-learning as shown in Equation 3.6.

$$Q^{new}_{s_v}(s_d, s_f) \leftarrow (1-\alpha) Q^{old}_{s_v}(s_d, s_f) + \alpha \left[R_t + \gamma * Q_{s_f}(s_d) \right] \tag{3.6}$$

The extent to which the newly learned Q-value affects the old Q-value is dependent on the learning rate, $\alpha(0,1]$. The closer the value of α is to one, the more the impact of the newly computed Q-value on the old one. If is equal to one, then the recently learned Q-value replaces the old Q-value completely.

The discount factor, $\gamma[0,1]$ controls the agent's liking for the future rewards with respect to the current reward. If γ is equal to 1, both the immediate reward and the future reward are considered equally.

The reward got when a sensor node, s_v chooses the next forwarder, s_f is modeled using that of RL-based Lifetime Optimization (RLLO) routing protocol [54], RL-Based Routing (RLBR) protocol [55], and RL for Lifetime Optimization (R2LTO) routing protocol[24] for WSNs.

The reward function of RLLO is given as:

$$R(s_v, s_f) = \frac{E_r(s_f)}{h(s_f)} \tag{3.7}$$

where $E_r(s_f)$ and $h(s_f)$ are the residual energy and hop count of the next forwarder respectively.

The reward function of RLBR is given as:

$$R(s_v, s_f) = \frac{E_r(s_f)}{d^n(s_v, s_f)h(s_f)} \tag{3.8}$$

where $d(s_v, s_f)$ is the link distance between the source node and the next forwarder and n is the path loss exponent. $d(s_v, s_f)$ and n is given in Equation 3.9 and Equation 3.10, respectively.

$$d(s_v, s_f) = \sqrt{(x(s_f) - x(s_v))^2 + (y(s_f) - y(s_v))^2} \tag{3.9}$$

where $(x(s_v), y(s_v))$ and $(x(s_f), y(s_f))$ are the position coordinates of the source node and next forwarder, respectively.

$$n = \begin{cases} 2, \, if \, d \leq d_o \\ 4, \, otherwise. \end{cases} \tag{3.10}$$

where d_o is known as the reference distance that is defined in Section 3.2.3

The reward function of R2LTO is given as:

$$R(s_v, s_f) = \frac{E_r(s_f)}{\frac{T_x(s_v, s_f)}{T_x(max)} h(s_f)} \tag{3.11}$$

where $T_x(s_v, s_f)$ is the estimation of the transmitted energy to send a packet from the source node to the next forwarder. $T_X(max)$ is the highest transmitted energy.

The Q-value of each sensor node, s_v is initialized using Equation 3.1 at the beginning of the learning round. Also, the path Q-value of each sensor node, s_v to its neighboring sensor node, s_k is initialized as zero for RLLO, RLBR, and R2LTO. At each learning round, the packet generated by each sensor node, s_v is routed to the sink by learning the optimal routing path, P. The first sensor node in the routing path, P is the source sensor node, s_v. Subsequently, for every neighboring sensor node, s_k of the source sensor node, s_v that is

not in the routing path, P of the source sensor node, s_v, the reward $R(s_v, s_k)$ is computed using Equation 3.7, Equation 3.8, and Equation 3.11 when dealing with RLLO, RLBR, and R2LTO, respectively. Also, the path quality of the source sensor node to its neighboring sensor node, $Q_{s_v}(s_v, s_k)$ is computed likewise using Equation 3.6. These path Q-values are then stored in the sensor node, s_v temporary path Q-values, TQ. The next forwarder, s_f of s_v is selected using the information contained in TQ with Equation 3.2. The next forwarder, s_f is added to the routing path, P. If s_f is not the sink, then s_f becomes the source node holding the packet. These processes are repeated until the next forwarder becomes the sink node. In the process of the learning round, if any sensor node depletes its energy source, the protocol is terminated. The algorithm of the RLLO, RLBR, and R2LTO is provided in Algorithm 6, Algorithm 7, and Algorithm 8, respectively.

Algorithm 6 RLLO

Input: $G(V, E)$, α, γ, ϵ, L
Output: Optimal Route

for $i = 1$ to L **do**
  Initialize $Q(s_v)\ \forall\ s_v \in S$ using Equation 3.1
  Initialize $Q_{s_v}(s_v, s_k) = 0\ \forall\ s_k \in Ng(s_v)$
  for $s_v \in S$ **do**
    $P = \{s_v\}$
    $TQ = \{\}$
    while True **do**
      for $s_k \in Ng(s_v)$ **do**
        if $s_k \notin P$ **then**
          if $d(s_v, s_s) \geq d(s_k, s_s)$ **then**
            Compute $R(s_v, s_k)$ using Equation 3.7
            Compute $Q_{s_v}(s_v, s_k)$ using using Equation 3.6
            $TQ \leftarrow Q_{s_v}(s_v, s_k)$
          end if
        end if
      end for
      Choose $s_f \in Ng(s_v)$ as next forwarder using Equation 3.2
      Update $Q(s_v)$ using Equation 3.4
      Update $h(s_v)$ using Equation 3.5
      $P \leftarrow s_f$
      if s_f is sink **then**
        Break
      end if
    end while
    if $E(s_v) = 0\ \forall\ s_v \in S$, **then**
      Break
    end if
  end for
end for

Algorithm 7 RLBR

Input: $G(V, E)$, α, γ, ϵ, L
Output: Optimal Route

for $i = 1$ to L **do**
  Initialize $Q(s_v)\ \forall\ s_v \in S$ using Equation 3.1
  Initialize $Q_{s_v}(s_v, s_k) = 0\ \forall\ s_k \in Ng(s_v)$
  for $s_v \in S$ **do**
    $P = \{s_v\}$
    $TQ = \{\}$
    while True **do**
      for $s_k \in Ng(s_v)$ **do**
        if $s_k \notin P$ **then**
          if $d(s_v, s_s) \geq d(s_k, s_s)$ and/or $h(s_v) \geq h(s_k)$ **then**
            Compute $R(s_v, s_k)$ using Equation 3.8
            Compute $Q_{s_v}(s_v, s_k)$ using using Equation 3.6
            $TQ \leftarrow Q_{s_v}(s_v, s_k)$
          end if
        end if
      end for
      Choose $s_f \in Ng(s_v)$ as next forwarder using Equation 3.2
      Update $Q(s_v)$ using Equation 3.4
      Update $h(s_v)$ using Equation 3.5
      $P \leftarrow s_f$
      if s_f is sink **then**
        Break
      end if
    end while
    if $E(s_v) = 0\ \forall\ s_v \in S$, **then**
      Break
    end if
  end for
end for

Algorithm 8 R2LTO

Input: $G(V, E)$, α, γ, ϵ, L
Output: Optimal Route

for $i = 1$ to L **do**
  Initialize $Q(s_v)\ \forall\ s_v \in S$ using Equation 3.1
  Initialize $Q_{s_v}(s_v, s_k) = 0\ \forall\ s_k \in Ng(s_v)$
  for $s_v \in S$ **do**
    $P = \{s_v\}$
    $TQ = \{\}$
    while True **do**
      for $s_k \in Ng(s_v)$ **do**
        if $s_k \notin P$ **then**
          Compute $R(s_v, s_k)$ using Equation 3.11
          Compute $Q_{s_v}(s_v, s_k)$ using using Equation 3.6
          $TQ \leftarrow Q_{s_v}(s_v, s_k)$
        end if
      end for
      Choose $s_f \in Ng(s_v)$ as next forwarder using Equation 3.2
      Update $Q(s_v)$ using Equation 3.4
      Update $h(s_v)$ using Equation 3.5
      $P \leftarrow s_f$
      if s_f is sink **then**
        Break
      end if
    end while
    if $E(s_v) = 0\ \forall\ s_v \in S$, **then**
      Break
    end if
  end for
end for

3.2.3 *Energy Consumption Model*

The energy consumed by the v^{th} sensor node, EC_{s_v} in a round of data transmission is the summation of the energy consumed by the sensor node in sending and receiving data packets and is given in the Equation 3.12 [63].

$$EC_{s_v}(p, d) = E_{tx}(p, d) + E_{rx}(p) \tag{3.12}$$

The energy consumed by a sensor node for sending and receiving data packets is given in Equation 3.13 and Equation 3.14, respectively [127].

$$E_{tx}(p, d) = \begin{cases} E_{elec}p\ell + E_{fs}pd^2 \ \ if\ d \leq d_o \\ E_{elec}p\ell + E_{mp}pd^4 \ \ if\ d > d_o \end{cases} \tag{3.13}$$

$$E_{rx}(p) = E_{elec}p\tau \tag{3.14}$$

where p is the number of bits per packet, d is the distance between the source node and the destination node, ℓ is the number of packets sent by a sensor node per round, τ is the number of packets received by a sensor node per round, $E_{tx}(p,d)$ is the transmitted energy, $E_{rx}(p)$ is the received energy, E_{elec} is the electronic energy consumed to transmit or receive unit data of the packet. E_{fs}, E_{mp} are the transmit amplifier efficiency and depend on the transmitter amplifier model (free space model is employed when $d \leq d_o$, otherwise the multipath model is employed). d_o is the baseline distance and is obtained by equating the two expressions of $E_{tx}(p,d)$ at $d = d_o$ and is given as:

$$d_o = \sqrt{\frac{E_{fs}}{E_{mp}}} \tag{3.15}$$

The energy consumption of a sensor node to send data packet(s) is a function of a continuous distance in Equation 3.13, but in practice, the transmission power of a sensor node at less than the transmission range is used. To implement this in the simulation the distance between two neighboring sensor nodes is used since the sensor nodes are only connected if the distance is less than or equal to the transmission radius.

3.3 SIMULATION AND RESULTS DISCUSSIONS

The performance analysis of the distributed RL-based lifetime-aware routing techniques is achieved by analyzing the impact of the learning rate and discount factor on the rate of convergence, network lifetime, and average energy consumption per round with simulations. This is to enable the right choice of the learning rate and discount factor to maximize the network lifetime and reduce average network energy consumption per round. The performance metric of the network lifetime and average network energy consumption per round of the RLLO, RLBR, and R2LTO are compared for network lifetime maximization. The network lifetime is computed as the number of rounds taken for the first sensor node to deplete its energy source. The average network energy consumption per round is the ratio between the total energy consumption of the sensor nodes and the network lifetime. The RLLO, RLBR, and R2LTO are implemented with Python 3.8 under the "PyCharm" development environment. The Python Networkx module [57] is used to implement the graphical structure of the WSN. The Python code is executed on the SLURM (Simple Linux Utility for Resource Management) cluster on the IRIT's OSIRIM platform. The Computer nodes of the OSIRIM platform adopted are the 4 AMD EPYC 7402 bi-processor computing nodes at 2.8 GHz, with 48 processors and 512 GB of RAM each. These nodes enable more than 24 threads and/or 192 GB of RAM for the same process. The simulation parameters used to implement the network and the deployed WSN are shown in Figure 3.1 and Table 3.1, respectively.

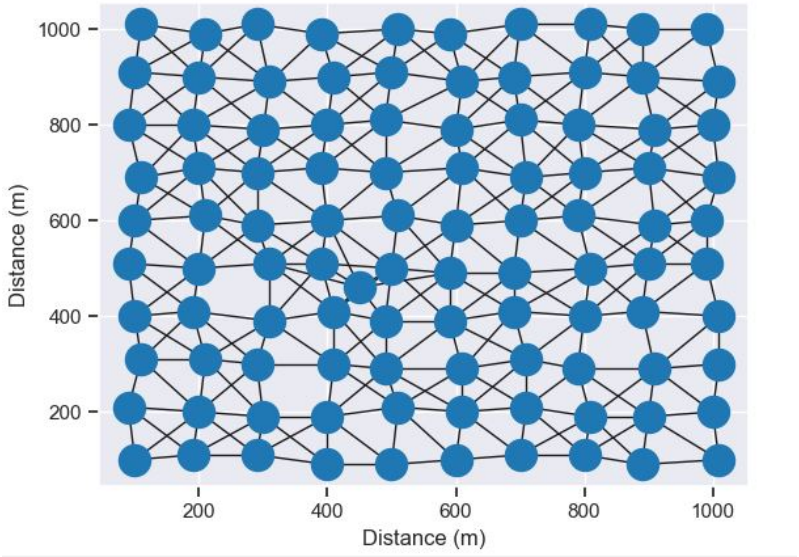

Figure 3.1: The Deployed WSN

Table 3.1: WSN Simulation Parameters

Parameters	**Values**
Number of sensor nodes	100
Number of sink node	1
Deployment area	1000 m × 1000 m
Sensor node deployment	Random
Sink position	$(500, 500)$
Transmission range	150 m
Data packet size	1024 bits
Packet generation rate	1 $/s$ to 10 $/s$
Initial energy of sensor nodes	10 J to 100 J
E_{elec}	50 nJ/bit [63]
e_{fs}	10 $pJ/bit/m^2$ [127]
e_{mp}	0.0013 $pJ/bit/m^4$ [127]

The impact of the learning rate on the network lifetime while the discount factor and epsilon are kept constant at 1 and 0.1, respectively of the distributed RL-based lifetime aware routing protocol considered which are RLLO, RLBR, and R2LTO is examined as shown in Figure 3.2.

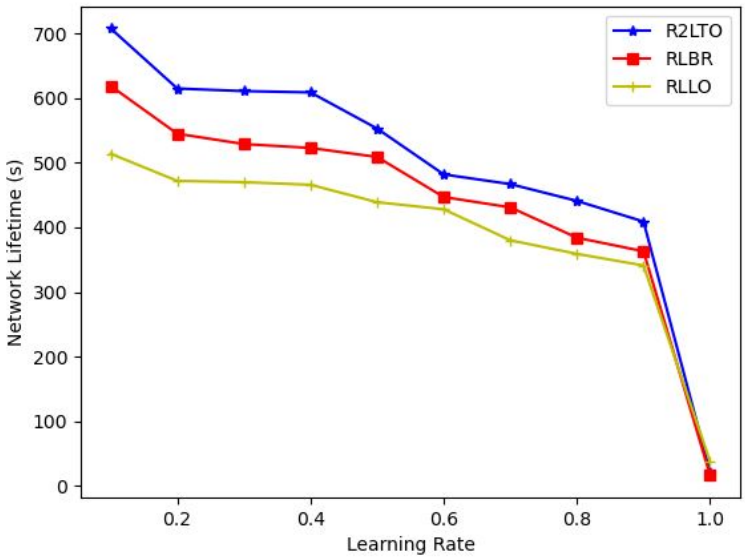

Figure 3.2: Network Lifetime of Distributed RL-based Lifetime Aware Routing Protocols with Learning Rate

It is seen in Figure 3.2 that as the learning rate of the protocols increases toward 1, the network lifetime decreases. This is because as the learning rate increases, the faster the tendency of the newly learned Q-value to replace the old learned Q-value. This makes the protocols lose the memory of the learned optimal routing paths for each sensor node to the sink. This causes a decrease in the network lifetime because there is an increase in the sensor nodes' energy consumption when non-optimal paths are used for routing. The increase in the network energy consumption as the learning rate increases makes the network have an increase in the average energy consumption per round as shown in Figure 3.3.

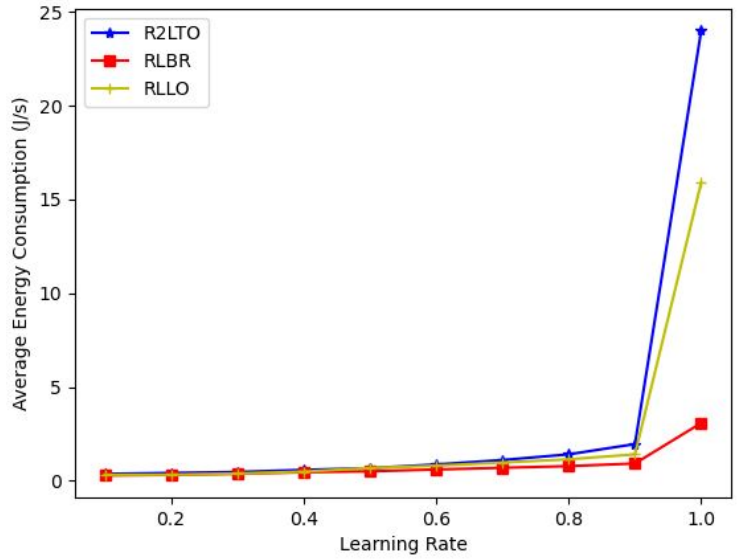

Figure 3.3: Average Energy Consumption of Distributed RL-based Lifetime Aware Routing Protocols with Learning Rate

Therefore, both protocols, have a better performance in network lifetime and average energy consumption per round when a lower learning rate (that is 0.1) is used.

Subsequently, the impact of the discount factor on the network lifetime for the RLLO, RLBR, and R2LTO is examined by varying the discount factor while the learning rate and epsilon are both kept at 0.1 as shown in Figure 3.4.

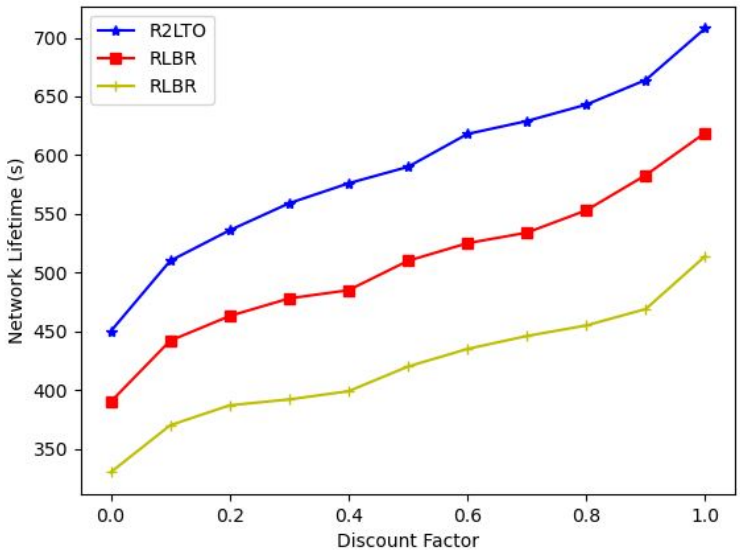

Figure 3.4: Network Lifetime of Distributed RL-based Lifetime Aware Routing Protocols with Discount Factor

It is seen in Figure 3.4 that as the discount factor of the protocols increases toward 1, the network lifetime increases. This is because as the discount factor increases, the immediate reward and future reward tend to be considered equally. This will make the learning process utilize the cumulative reward. Thus the learning agent will choose the best routing paths, resulting in an increased network lifetime because of reduced energy consumed by the sensor nodes in using optimal routing paths in sending data packets to their next forwarder.

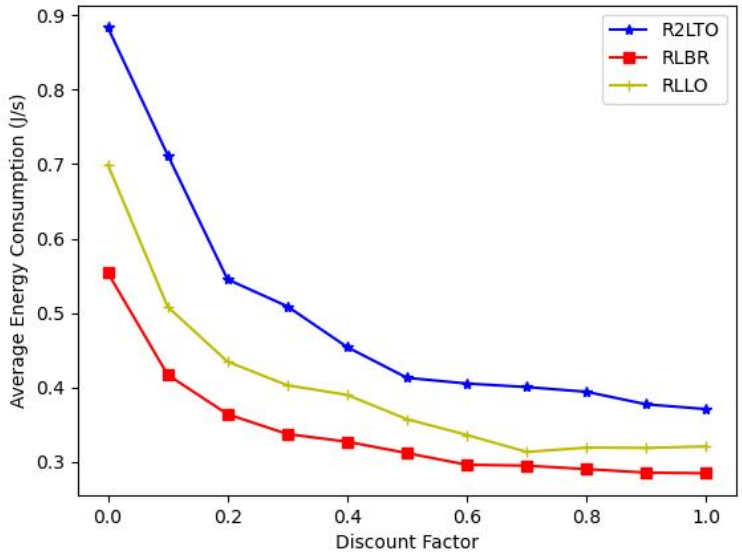

Figure 3.5: Average Energy Consumption of Distributed RL-based Lifetime Aware Routing Protocols with Discount Factor

The decrease in the network energy consumption as the discount factor increases make the network have a decreasing average energy consumption per round as shown in Figure 3.5.

Therefore, both protocols, have a better performance in network lifetime and average energy consumption per round when a higher discount factor (1) is used.

The convergence of the normalized average Q-value with the network lifetime for both protocols when the learning rate is 0.1 and the discount factor is 1 is shown in Figure 3.6.

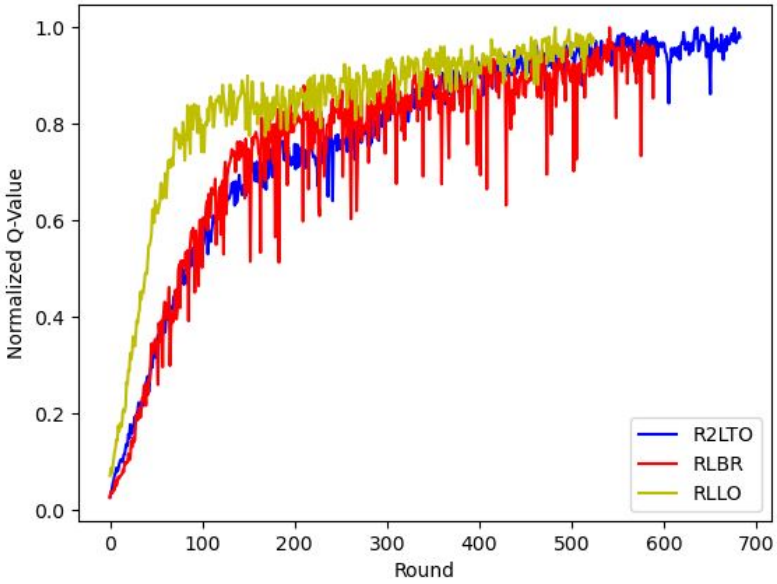

Figure 3.6: Normalized Average Q-value of Distributed RL-based Lifetime Aware Routing Protocols with Learning Round

As seen in Figure 3.6 the convergence rate of the protocols is low because each sensor node learns the next forwarder locally and needs many iterations to select the optimal next forwarder, this leads to the degradation of the network lifetime and average energy consumption per round.

3.4 CONCLUSION

Performance analysis of the impact of Q-learning parameters for distributed RL-based routing protocols to improve the WSNs network lifetime and the energy consumption is presented in this chapter. The lifetime and energy consumption is better for a low learning rate and high discount factor. This resulted in an improved lifetime and average energy consumption per round of the distributed RL-based lifetime-aware routing protocols. The lower the learning rate, the better the preservation of the learned optimal routing path. The higher the discount factor, the more the learning process utilizes the cumulative reward. Thus the learning agent will choose the best routing paths which will result in an increased network lifetime because of reduced energy consumed by the sensor nodes in sending data packets to their next forwarder. However, the convergence rate of the distributed RL-based lifetime-aware routing protocols is low because each sensor node learns the next forwarder locally and needs many iterations to select the optimal next forwarder, this leads to the degradation of the network lifetime and average energy consumption per round. In the next chapter, a novel solution in the case of a centralized RL-based lifetime-aware approach will be provided.

4

LIFETIME-AWARE CENTRALIZED ROUTING PROTOCOL FOR WIRELESS SENSOR NETWORKS USING Q-LEARNING

4.1 INTRODUCTION

This chapter presents the design of a Lifetime-Aware Centralized Q-routing Protocol (LACQRP) for WSN to maximize the network lifetime. This is achieved by implementing Q-learning on the sink of the WSN, which also acts as a controller that has global knowledge of the network topology. The controller generates all possible distance-based minimum spanning trees (MSTs), which form the set of routing tables (RTs). The maximization of the network lifetime is achieved by the controller learning the routing table that minimizes the maximum of the sensor nodes' consumption energies using Q-Learning.

4.2 METHODOLOGY

This section presents the network model, the implementation of the All-MSTs algorithm, and the designing and implementation of the LACQRP.

4.2.1 *Network Model*

The topology of the WSN is modeled as a weighted graph, $G = (V, E)$. V is the set of network nodes (vertices) and E is the set of network links (edges). The connection between two nodes in the network is represented by a distance edge weight. In the proposed routing protocol, each WSN node broadcasts *Hello* packets after the initialization of the network. Based on the received *Hello* packets, the sink/controller builds the network graph and computes a list of all routing tables (RTs) based on distance-based Minimum Spanning Trees (MSTs) [144]. The choice of the distance-based MST is because the transmission energy that the sensor nodes use to send packets is a function of the distance between nodes.

4.2.2 *All-MSTs Algorithm*

The algorithm and the complexity of generating all MSTs are explained in the sequel.

The node set and edge set of the network graph are $V = \{v_1, ..., v_n\}$ and $E = \{e_1, ..., e_m\} \subseteq V \times V$, respectively. An integer weight $w(e) > 0$ is associated with each edge $e \in E$. The sum of the weights of constituent edges for an MST is depicted as weight $w(T)$. The list of all MSTs of the network graph is obtained by using a set of fixed edges $F = \{e_1, ..., e_k\}$ and a set of restricted edges $R \subseteq E$ in G that is disjoint with F, where k is the number of elements in F.

An MST is said to be (F, R)-admissible if it contains all edges of F, but does not contain those of R. An MST of G, obtained by any standard MST algorithm [44] is used to divide the problem P of finding an MST into a set of mutually disjoint sub-problems $P(\{e_1, ..., e_{i-1}\}, \{e_i\})$, where $i = 1, ..., n-1$. This implies that the sub-problems

$i = 1, ..., n-1$ list all the MSTs that contain $e_1, ..., e_{i-1}$, but do not contain e_i. Therefore the problem P becomes $P(F, R)$: List all the MSTs, which are (F, R)-admissible.

An MST that is (F, R)-admissible is denoted by $T(F, R) = F \cup \{e_{k+1}, ..., e_{n-1}\}$. For $i = k+1, ..., n-1$, F_i and R_i are defined as $F_i = F \cup \{e_{k+1}, ..., e_{n-1}\}$ and $R_i = R \cup \{e_i\}$, respectively. Moreover, let e be an arbitrary edge of an (F, R)-admissible MST, T of G. Deleting e from T divides it into two non-connected components V_1 and V_2. $Cut(e)$ is the set of edges that can substitute e and reconnect V_1 and V_2, and is defined as $Cut(\hat{e}) = \{e^* \in E | e^* \in (V_1 \times V_2) \cup (V_2 \times V_1)\}$.

From the cut-set optimality condition for MST, for a pair of edges $e \in T$ and $e^* \in Cut(e) \backslash \{e\}$, $T \cup \{e^*\} \backslash \{e\}$ defines an MST. Renumbering the vertices of the (F, R)-admissible MST at each sub-problem $P(F, R)$ in a post-order fashion as T is transverse from an arbitrary root as $\{v_i | i = 1, ..., n\}$, this makes T to be rooted at v_n, thereby making e_i to be an edge connecting v_i to be its parent vertex in T. An interval $[\sigma_i, \psi_i]$ is associated with v_i and represents the set of descendants of it. This implies that $j \in [\sigma_i, \psi_i] \iff v_j$ is a descendant of v_i in T rooted at v_n.

Denoting $E_i = \{(v_i, v_j) \in E | (v_i, v_j) \notin T\}$ as a set that is not tree edges incident on vertex v_i and the set of quasi-cuts, Q to be a set of elements of the form $(w, v, v^*) \in Q$. This implies $e = (v, v^*) \in E$ has weight $w(e) = w$ and is a candidate of a cut-set edge. Q is then use to find the substitute e_i^* of $e_i \in T \backslash F$ which enables getting $T \cup \{e_i^*\} \backslash \{e_i\}$ as a new MST and updating it for the next i^{th} sub-problem.

The algorithm for generating all the possible MSTs of a network graph using this description is given in Algorithm 9.

Algorithm 9 : All-MSTs Algorithm

Input: F, R, T
Output: All MSTs

```
Q = {}
for i = 1 to n − 1 do
  for e = (v^i, v^j) ∈ E^i \ R do
    if j < σ_i then
      Reverse the direction
      if (w(e), j, i) ∈ Q then
        Delete it from Q
        Insert (w(e), i, j) into Q
      end if
    end if
    if j ∈ [σ_i, ψ_i] then
      if (w(e), j, i) ∈ Q then
        Delete it from Q
      end if
    end if
    if j > σ_i then
      Insert (w(e), i, j) into Q
    end if
  end for
  if e_i ∉ F then
    Find (w, i*, j*) ∋ w = w(e_i) and j* ∈ [σ_i, ψ_i]
    if such an (w, i*, j*) is found with j* ∈ [σ_i, ψ_i] then
      Delete (w, i*, j*) from Q, and go to line 21
    end if
    if such an (w, i*, j*) is found with j* ∉ [σ_i, ψ_i] then
      Set e_i* = (v^{i*}, v^{j*}). {Subsitute for e_i found.}
    end if
  end if
end for
for i = k + 1 to n − 1 do
  if e_i* exists, then
    Set T_i = T ∪ {e_i*} \ {e_i}
    Output T_i {Comment: A new MST is found}
    Set F_i = F ∪ {e_{k+1}, ..., e_{i−1}} and R_i = R ∪ {e_i}
    Call All MST(F_i, R_i, T_i) recursively
  end if
end for
```

The All-MSTs algorithm runs in $O(Nm \log n)$ time and $O(m)$ space. Where n, m, and N are the number of nodes, edges, and MSTs of the network graph, respectively. This is because Q includes a maximum of m elements, and for every non-tree edge, a maximum of two *Inserts* and two *Deletes* are performed. Also, for every tree edge, a maximum of one *Find* is performed. Because Q is an order set, every *Inserts*, *Deletes*, and *Find* is executed in $O(\log m)$ time. Therefore the total substitutes are performed in $O(m \log m) = O(m \log n)$ time. The other computation like traversing G along T, finding intervals $[\sigma_i, \psi_i]$, and renumbering V in post-order manner is done in $O(m)$ time.

4.2.3 *Lifetime-Aware Centralized Q-Routing Protocol*

The lifetime of the network is considered as the time required for the first sensor node to die. Therefore, a centralized RL-based unicast routing protocol is designed for a WSN to maximize the minimum Estimated Node Residual Energy $(ENRE)$ of the sensor nodes in the network. Therefore, the optimization problem is to find the MST of the WSN such that:

$$Minimum\ ENRE_n\ (for\ all\ n)\ is\ Maximized \tag{4.1}$$

This is because, despite using the MSTs as the routing tables (RTs) to minimize the energy consumption of sensor nodes, the number of paths crossing each sensor node in the different RTs differs. This parameter makes the energy consumption of each sensor node to be different when using the different RTs for routing. The RT that has the minimum number of paths crossing a particular sensor node will drain less energy from the sensor node.

Therefore, to prolong the time taken for the first sensor node to die, the proposed routing protocol tends to find the RT that has the least number of paths crossing a particular sensor node to be used for routing packets to the sink.

The learning agent is located at the controller which also acts as the sink. The sink collects all the data sent by the sensor nodes in the network. The controller builds all the possible RTs. Therefore, the state space S and action space A of the learning agent are the lists of all RTs. The state of the learning agent is the RT that the sink is using to receive packets from the sensor node at the current learning round and the action is to choose the next RT that will optimize the network lifetime.

After, each round of data transmission by the sensor nodes to the sink, each sensor node sends its residual energy to the sink. Based on the residual energy of each sensor node, the sink estimates the energy consumption of each sensor node in the previous round of data transmission.

To make the learning meaningful, the Q-value in Equation 4.2 is made to denote the value of the maximum energy consumption of the sensor nodes in the network when using a particular RT in sending data packets to the sink.

$$Q_t(s_t, a_t) = (1-\alpha)Q_{t-1}(s_t, a_t) + \alpha\left[R_t + \gamma * \max_{a\in A}\{Q(s_{t+1}, a)\}\right] \tag{4.2}$$

where α is the learning rate and γ is the discount factor.

The achievable reward R_t in each learning round is modeled as the maximum of the energy consumption by the sensor nodes in the network when a particular RT is used and is given as:

$$R_t = \max_{n\in V}\{EC_n\} \tag{4.3}$$

where EC_n is the energy consumption of the n^{th} sensor node and V is the set of sensor nodes in the WSN.

This implies that the sink evaluates the effectiveness of the RT based on the reward function after a round of data transmission. To maximize the minimum estimated node residual energy of the sensor nodes and thereby maximize the lifetime of the WSN, exploration of the solution search space is ensured by choosing the RT with a minimum Q-value using the epsilon-greedy strategy [131].

That is given a probability value of epsilon, $\epsilon \in [0,1]$ and a random number, $r \in (0,1)$ generated in each learning round, the action a_t is selected as:

$$a_t = \begin{cases} Random\ action,\ if\ r \geq 1-\epsilon \\ \underset{a \in A}{\operatorname{argmin}}\{Q_t(s,a)\},\ otherwise. \end{cases} \tag{4.4}$$

Because the Q-matrix of the learning agent scales with the state space and action space. Also, Q-learning performance depends on the reward function and how the action space is scanned. The Q-value for each state-action pair is initialized as the maximum initial residual energy of the sensor nodes, E_{max}. This will enable fast convergence to the optimal RT(s) with an epsilon-greedy strategy.

The proposed LACQRP for finding the optimal RT for maximizing the network lifetime of the WSN is given in Algorithm 10.

Algorithm 10 : LACQRP

Input: Learning rate, Discount factor, Epsilon, Number of learning rounds, List of RTs
Output: Optimal RT

1: Initialize the Q-value for each state-action pair as E_{max}.
2: Initialize a random RT as the current state of the sink.
3: **for** $i = 1$ to Number of learning rounds **do**
4: The sink chooses an RT using Equation 4.4 and broadcasts it to each sensor node.
5: Each sensor node sends data to the sink using the RT.
6: The sink evaluates the effectiveness of the RT using Equation 4.3.
7: The sink updates its Q-value using Equation 4.2.
8: The sink updates its state as the current RT.
9: **if** Any sensor node depletes its energy source, **then**
10: *break*
11: **end if**
12: **end for**
13: **return** The optimal RT is the RT with the highest percentage utilization.

Since Algorithm 10 depends on Algorithm 9 to generate all MSTs, which are used as the RTs, the asymptotic time complexity of Algorithm 10 is the same as that of Algorithm 9.

Also, Algorithm 10 requires the initialization of a Q-matrix which depends on the size of the list of RTs. Therefore the asymptotic space complexity of Algorithm 10 is $O(N)$. Where the upper bound of N is $n^{(n-2)}$.

The convergence of the LACQRP to the optimal RT will be demonstrated by simulation. The optimal RT is the RT that has the highest percentage utilization. The percentage utilization of an RT is the ratio between the time the RT is used and the network lifetime. That is, the percentage utilization of an RT is given as:

$$U_{RT} = \frac{T_{RT}}{LT} \tag{4.5}$$

where U_{RT} is the percentage utilization of an RT, T_{RT} is the time the RT is used, and LT is the network lifetime.

4.3 SIMULATION AND RESULTS DISCUSSIONS

The performance of the LACQRP is achieved by simulations using the performance metric of convergence rate, network lifetime, and average energy consumption. The performance analysis of the LACQRP is achieved first by analyzing the impact of the learning rate and discount factor on the rate of convergence, network lifetime, and average energy consumption per round. This enables the right choice of the learning rate and discount factor to maximize the network lifetime and reduce average network energy consumption per round. The convergence rate, network lifetime, and average energy consumption of the LACQRP are compared with two distributed RL protocols for network lifetime maximization for WSN which are RLBR [55] and R2LTO [24]. The network lifetime is computed as the time taken for the first sensor node to deplete its energy source. The average energy consumption is computed as the total network energy consumption divided by the network lifetime. The energy consumption of the n^{th} sensor node in each round is the difference between its previously estimated node residual energy, $ENRE_n^{Previous}$ and its currently estimated node residual energy, $ENRE_n^{Current}$ after the end of a round. Therefore, the energy consumption of the n^{th} sensor node after a learning round is given as:

$$EC_n = ENRE_n^{Previous} - ENRE_n^{Current} \tag{4.6}$$

EC_n is the energy consumption of the n^{th} sensor node after a round. The energy consumed in sending and receiving data is computed by the energy model provided in Section 3.2.3

The simulation parameters used to implement the network and the randomly generated connected WSN are the same as in Figure 3.1. and Table 3.1, respectively.

4.3.1 *Impact of Learning Rate on LACQRP Network Lifetime and Average Energy Consumption*

The impact of the learning rate on the LACQRP network lifetime while the discount factor and epsilon are kept constant arbitrarily at 0 and 0.1 respectively, while the initial residual energy and packet generation rate of sensor nodes are $100J$ and $1/s$, respectively is examined as shown in Figure 4.1.

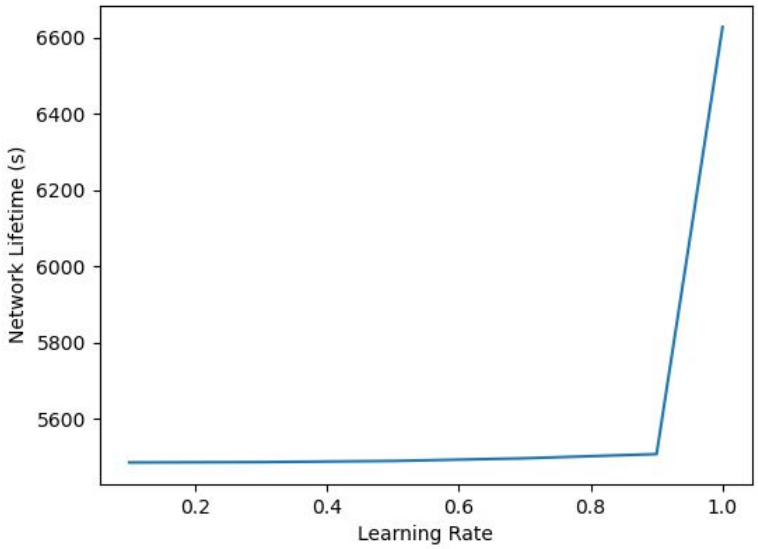

Figure 4.1: Network Lifetime of LACQRP with Learning Rate

It is seen that as the learning rate of the protocols increases toward 1, the network lifetime increases. This is because as the learning rate increases, the faster the tendency of the LACQRP to learn the optimal MSTs that maximize the network lifetime from the large solution search space of the LACQRP due to the utilization of all MSTs of the network graph. This causes an increase in the network lifetime because there is a decrease in the sensor nodes' energy consumption when optimal paths are used for routing. The decrease in the network energy consumption as the learning rate increases makes the network have a decrease in the average energy consumption per round as shown in Figure 4.2.

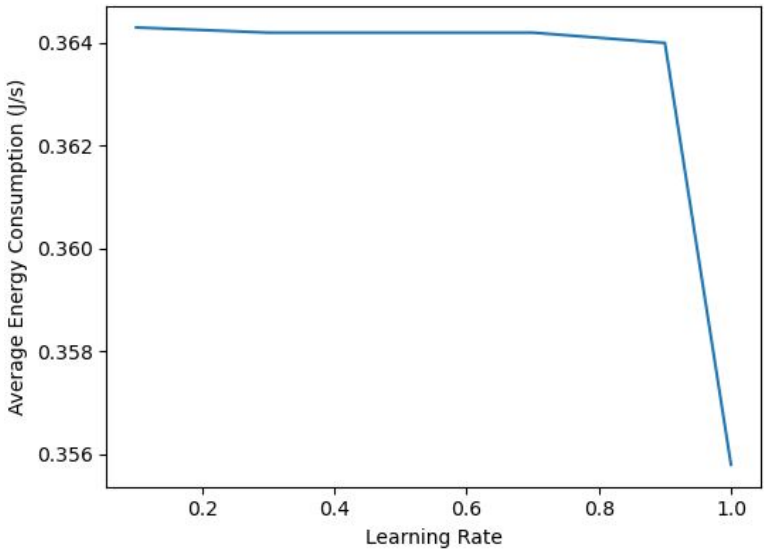

Figure 4.2: Average Energy Consumption of LACQRP with Learning Rate

Therefore, LACQRP has a better performance in network lifetime and average energy consumption per round when a higher learning rate (that is 1) is used.

4.3.2 *Impact of Discount Factor on LACQRP Network Lifetime and Average Energy Consumption*

Subsequently, the impact of the discount factor on the LACQRP network lifetime while the learning rate and epsilon are kept constant at 1 and 0.1 respectively, while the initial residual energy (IRE) and packet generation rate (PGR) of sensor nodes are $100J$ and $1/s$, respectively is examined as shown in Figure 4.3.

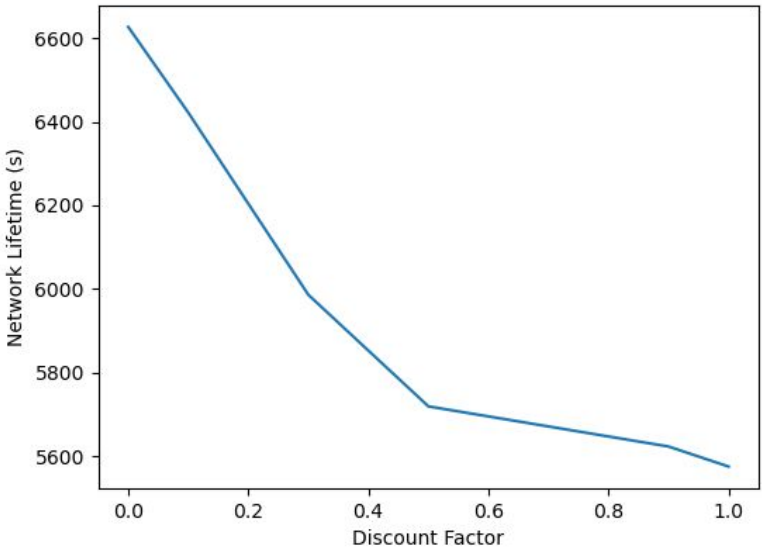

Figure 4.3: Network Lifetime of LACQRP with Discount Factor

It is seen that as the discount factor of the protocols increases toward 1, the network lifetime decreases. This is because as the discount factor increases, the immediate reward and future reward tend to be considered equally. This will make the learning process utilize the cumulative reward. This will cause the non-convergence of the Q-values of the learning agent and a consequent non-convergence of the optimal MSTs that maximize the network lifetime. Thus the learning agent will not choose the best routing paths, resulting in a decreased network lifetime because of increased energy consumed by the sensor nodes in using non-optimal routing paths in sending data packets to sink. The increase in the network energy consumption as the discount factor increases make the network have an increasing average energy consumption per round as shown in Figure 4.4.

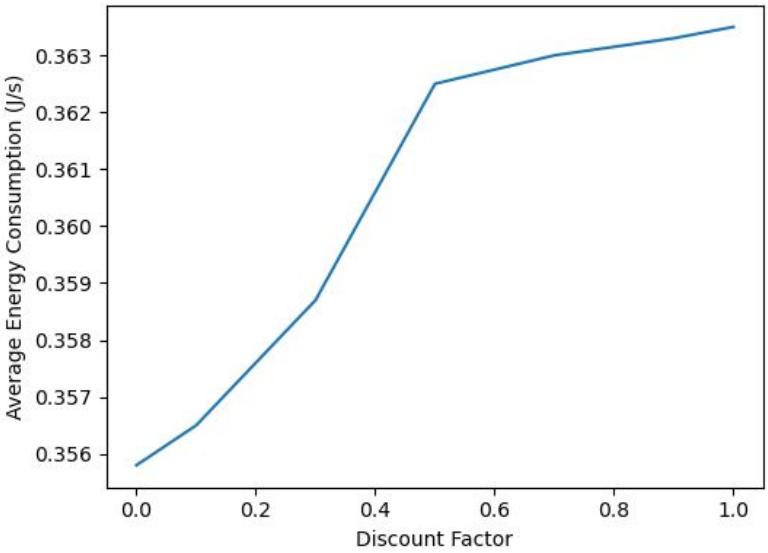

Figure 4.4: Average Energy Consumption of LACQRP with Discount Factor

Therefore, LACQRP has a better performance in network lifetime and average energy consumption per round when a low discount factor (0) is used.

4.3.3 *Performance Comparison of LACQRP with RLBR and R2LTO for Homogeneous Sensor Nodes IRE and PGR*

The LACQRP learns the optimal MST with the highest percentage utilization that maximizes the network lifetime by comparing its Q-value convergence rate with that of RLBR and R2LTO for the different rounds of data transmission as shown in Figure 4.5. The Q-value convergence rate of LACQRP approaches one faster when compared with that of RLBR and R2LTO. This is because LACQRP is centralized and has global knowledge of the network information and learns the optimal MST that maximizes the network lifetime. The maximum normalized Q-value of 1 for the LACQRP represents the optimal MST that maximizes the network lifetime.

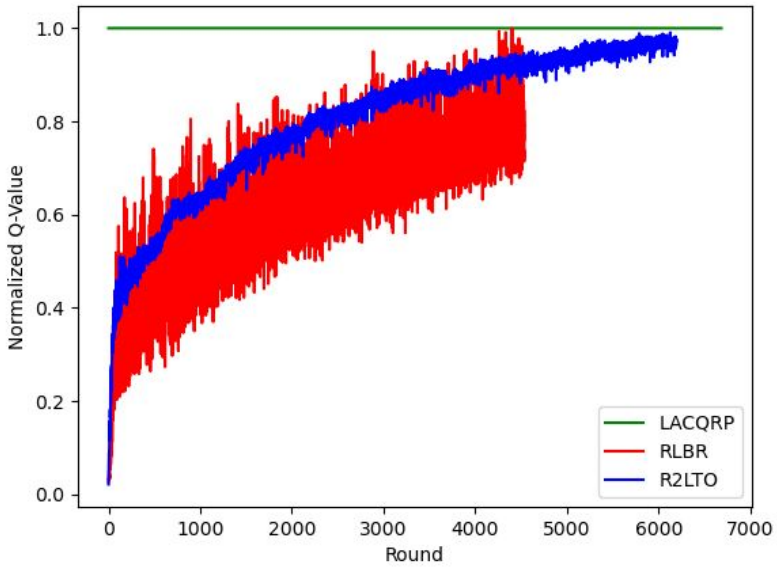

Figure 4.5: Q-Value Convergence Rate with Round of Data Transmission

Figure 4.6 shows the network lifetime of both protocols when the packet generation rate is set as one packet per second and the initial sensor node's energy is varied with the same amount. The network lifetime increases with the initial sensor node's residual energy. This is because the lifetime of a sensor node is proportional to its residual energy. The LACQRP has a better network lifetime performance of 46.34% and 11.28% when compared to RLBR and R2LTO, respectively with increasing sensor node residual energy.

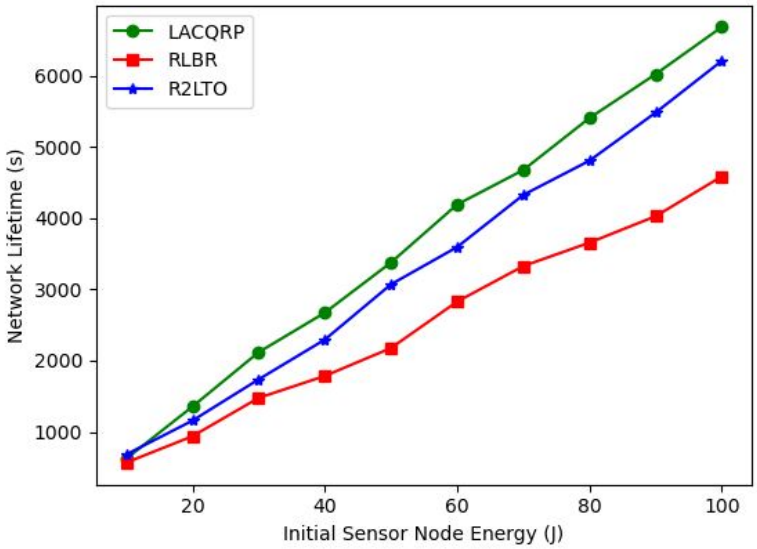

Figure 4.6: Network Lifetime with Increasing Initial Sensor Node Energy

This is because the LACQRP agent has global information on the network topology and can quickly learn the best MST from the list of all MSTs that maximizes the network lifetime by balancing the energy consumption among sensor nodes and minimizes average energy consumption as shown in Figure 4.7. LACQRP is different from the RLBR and R2LTO which are distributed in nature and are constrained by the learning agent having local information for the entire network. This results in a delay in learning the optimal routing path and therefore degrades the network lifetime. Both RLBR and R2LTO consider the energy of sensor nodes, their corresponding distances, and hop counts to the sink in choosing the next forwarder. Subsequently, RLBR does not select a next forwarder that has a greater distance or hops count to the sink when compared to the current node. This makes the network lifetime of R2LTO to be higher than that of the RLBR at the expense of increased energy consumption.

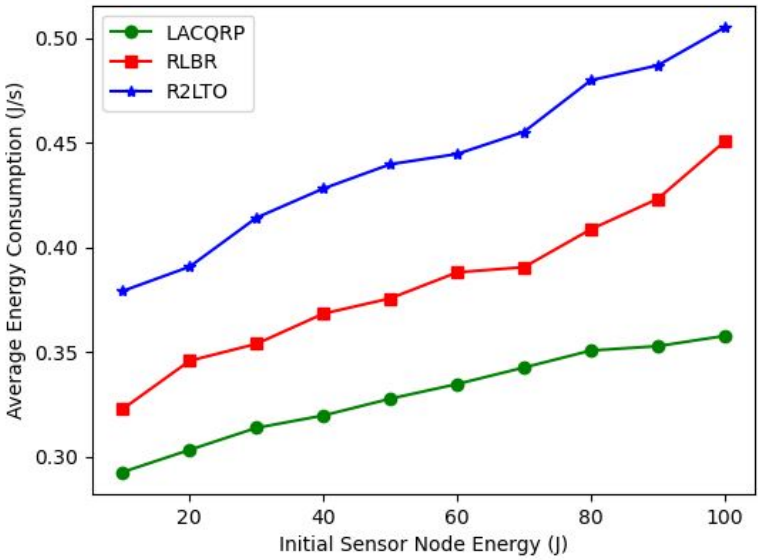

Figure 4.7: Average Energy Consumption with Increasing Initial Sensor Node Energy

Subsequently, The network energy consumption for both protocols increases with the increase of the initial energy of the sensor nodes. This is because the more the initial sensor nodes' energy, the more rounds of data transmission it takes for the sensor nodes to deplete their energy. LACQRP has a reduced average energy consumption of 13.88% and 25.47% when compared with RLBR and R2LTO, respectively. The reduction of the energy consumption of the LACQRP is because it utilizes the distance-based MSTs and network energy consumption decreases with minimum distance between sensor nodes. Also, LACQRP learns the optimal MST that minimizes the sensor nodes' energy consumption globally since the sink has global network information and can find the optimal MSTs to be used by all sensor nodes to send the data packet to the sink so that the network energy consumption is minimized and the network lifetime maximized.

However, the performance improvement of the network lifetime and average energy consumption of LACQRP, when compared with RLBR and R2LTO when the packet generation rate of the sensor node is constant and the initial residual energy is increased, comes with an increased computation time of 843% and 767% when compared with RLBR and R2LTO, respectively as shown in Figure 4.8. This is because of the NP-hardness of generating all MSTs by the LACQRP.

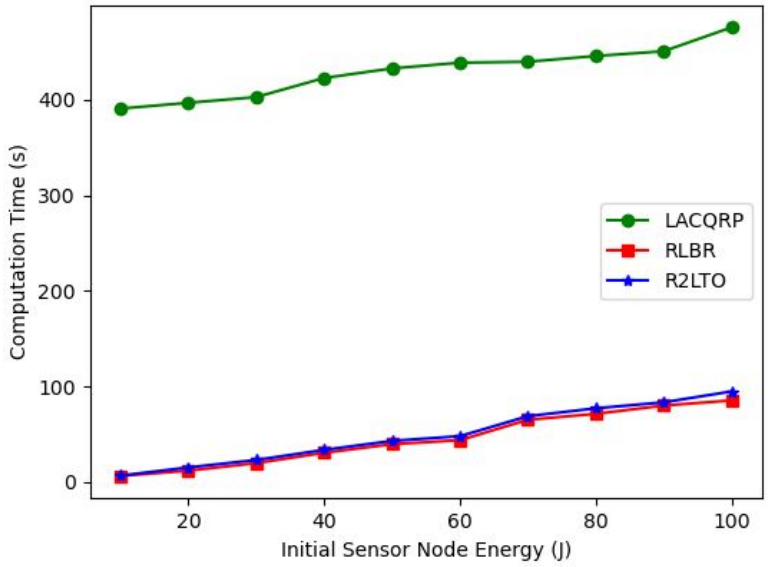

Figure 4.8: Computation Time with Increasing Initial Sensor Node Energy

The network lifetime of LACQRP is also compared with RLBR and R2LTO when the initial sensor node energy is set as 100 J for increasing packet generation rate as shown in Figure 4.9.

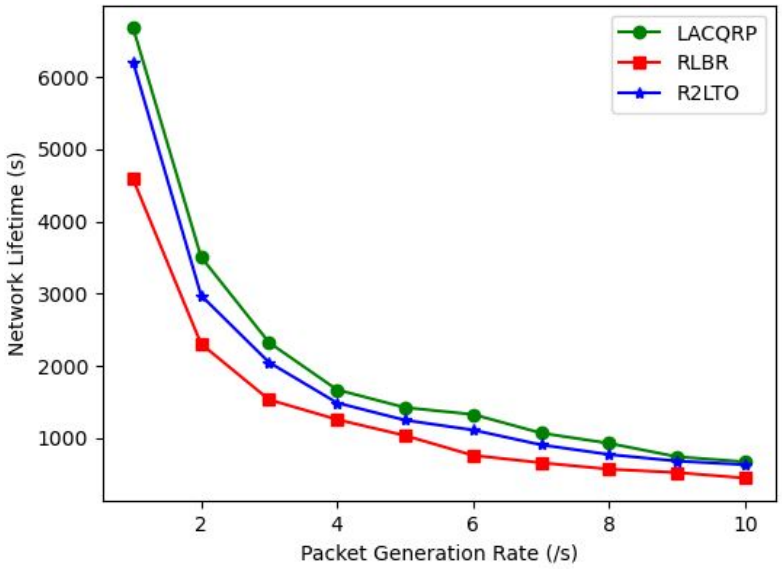

Figure 4.9: Network Lifetime with Increasing Packet Generation Rate

The network lifetime of both protocols decreases as the packet generation rate increases. This is because the energy consumption of the sensor nodes increases as the packet generation rate increases. This will subsequently lead to the first sensor node depleting its energy source on time. The LACQRP has a better network lifetime performance of 48.67% and 12.55% when compared to RLBR and R2LTO, respectively with increasing packet generation rates at the sensor nodes. This is because the LACQRP convergences quickly to the optimal MST that minimizes the sensor nodes' energy consumption as shown in Figure 4.10. This is against the distributed routing protocols of RLBR and R2LTO that require more time to converge to the optimal routing path.

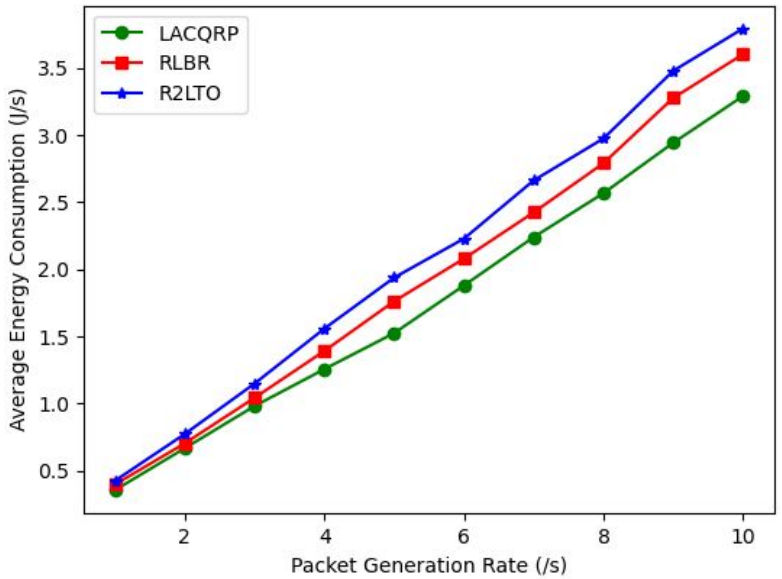

Figure 4.10: Average Energy Consumption with Increasing Packet Generation Rate

Subsequently, The network energy consumption for both protocols increases with the increase of the packet generation rate of the sensor nodes. This is because the more the sensor nodes' packet generation rate, the more the sensor nodes deplete their

energy. LACQRP has a reduced average energy consumption of 9.07% and 15.61% when compared with RLBR and R2LTO, respectively. The reduction of the energy consumption of the LACQRP is because of the utilization of the distance-based MST and can learn the optimal MST that minimizes the sensor nodes' energy consumption globally since the sink has global network information and can find the optimal MSTs to be used by all sensor nodes to send the data packet to the sink so that the network energy consumption is minimized and the network lifetime maximized.

However, the performance improvement of the network lifetime and average energy consumption of LACQRP, when compared with RLBR and R2LTO when the packet generation rate of the sensor node is increasing and the initial residual energy is constant, comes with an increased computation time of 921% and 887% when compared with RLBR and R2LTO, respectively as shown in Figure 4.11. This is because of the NP-hardness of generating all MSTs by the LACQRP.

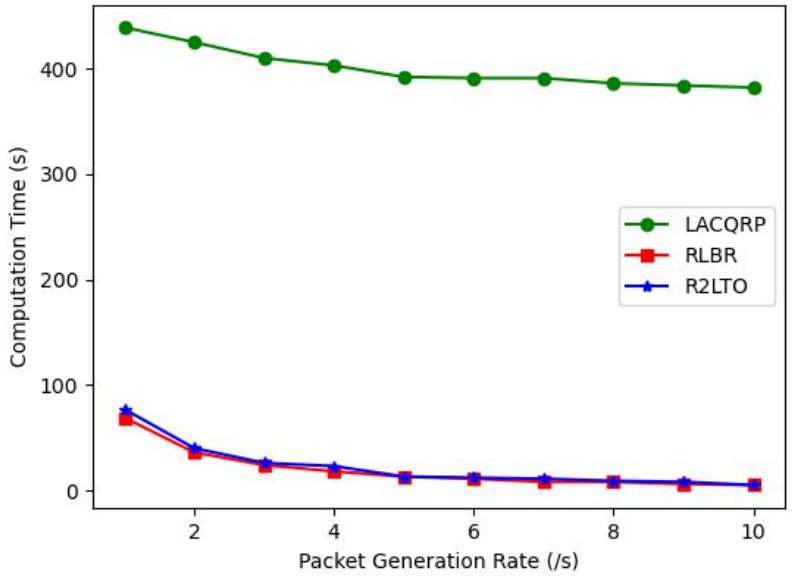

Figure 4.11: Computation Time with Increasing Packet Generation Rate

4.3.4 *Performance Comparison of LACQRP with RLBR and R2LTO for Heterogeneous IRE and Homogeonous PGR of Sensor Nodes*

The network lifetime and average energy consumption of LACQRP are also compared with that of RLBR and R2LTO when the energy of the sensor nodes is different and the packet generation rate is increased. The sensor nodes' energy is different by assigning the energy value from the range of 10J and 100J at the multiples of 10J. The packet generation rate of the sensor nodes is then constantly increased from 1/s to 10/s at the multiples of 1/s. For variable energy of the sensor nodes, LACQRP learns the optimal MSTs that balance the energy consumption of the sensor nodes so as to maximize the network lifetime. The Q-value convergence rate of LACQRP for variable sensor nodes' energy with that of RLBR and R2LTO for the different rounds of data transmission is shown in Figure 4.12. The Q-value convergence rate of LACQRP approaches one faster when compared with that of RLBR and R2LTO. This is because LACQRP is centralized and has global knowledge of the network information and learns the optimal MSTs that maximize the network lifetime.

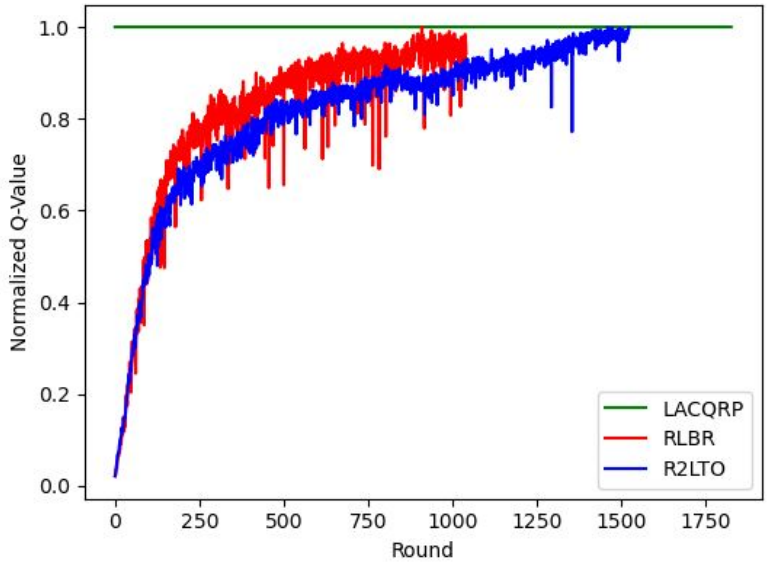

Figure 4.12: Q-Value Convergence Rate with Increasing Packet Generation Rate for Variable Sensor Node Energy

The network lifetime of both protocols decreases as the packet generation rate increases as shown in Figure 4.13. Consequently, LACQRP has an improved network lifetime performance of 61.28% and 20.29% when compared with RLBR and R2LTO, respectively when the energy of the sensor nodes is different and the packet generation rate of the sensor nodes is increased.

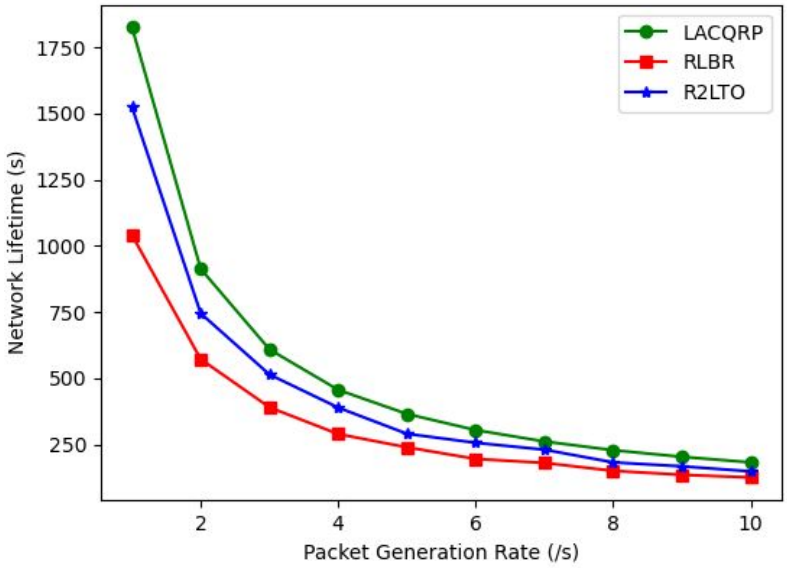

Figure 4.13: Network Lifetime with Increasing Packet Generation Rate for Variable Sensor Node Energy

Likewise, the average energy consumption of the protocols increases as the packet generation rate at the sensor nodes increases as shown in Figure 4.14. LACQRP has an improved average energy consumption performance of 9.27% and 19.34% when compared with RLBR and R2LTO, respectively when the energy of the sensor nodes is different and the packet generation rate of the sensor nodes is increased.

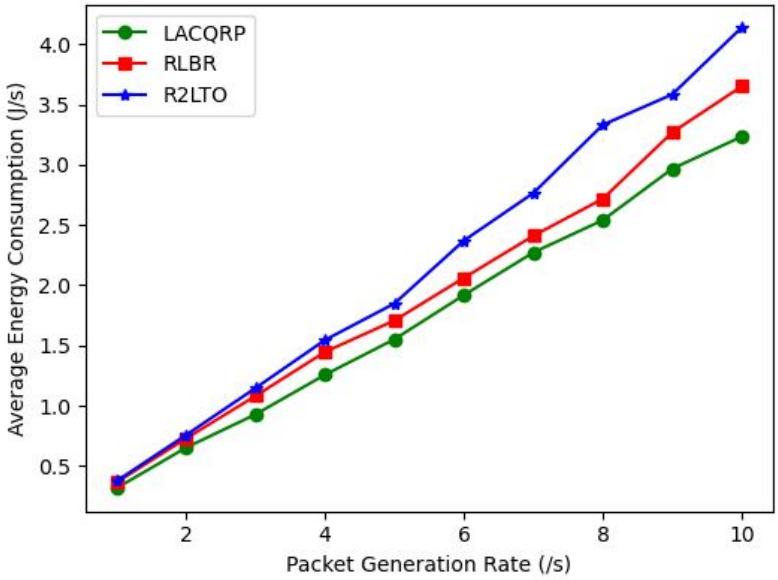

Figure 4.14: Average Energy Consumption with Increasing Packet Generation Rate for Variable Sensor Node Energy

However, the performance improvement of the network lifetime and average energy consumption of LACQRP, when compared with RLBR and R2LTO when the packet generation rate of the sensor node is increasing and the initial residual energy of the sensor nodes is heterogenous, comes with a very high computation time when compared with RLBR and R2LTO, respectively as shown in Figure 4.15. This is because of the NP-hardness of generating all MSTs by the LACQRP.

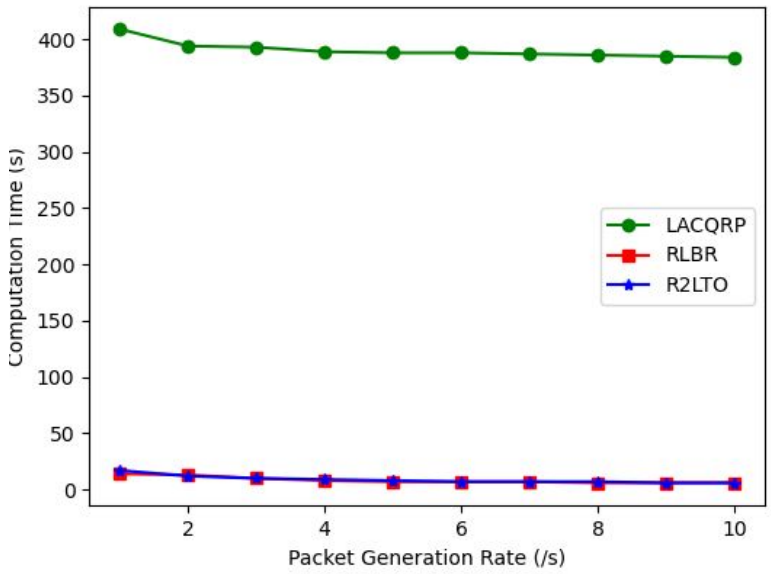

Figure 4.15: Computation Time with Increasing Packet Generation Rate for Variable Sensor Node Energy

4.3.5 *Performance Comparison of LACQRP with RLBR and R2LTO for Heterogeneous PGR and Homogeonous IRE of Sensor Nodes*

The network lifetime and average energy consumption of LACQRP are also compared with that of RLBR and R2LTO when the packet generation rate of the sensor nodes is different and the sensor nodes' residual energy is increased. The sensor nodes' packet generation rate is different by assigning the packet generation rate value from the range of $1/s$ to $10/s$ at the multiples of $1/s$. The initial residual energy of the sensor nodes is then constantly increased from $10J$ to $100J$ at the multiples of 10J. For the variable packet generation rate of the sensor nodes, LACQRP learns the optimal MSTs that balance the energy consumption of the sensor nodes so as to maximize the network lifetime. The Q-value convergence rate of LACQRP for variable sensor nodes' packet generation rate with that of RLBR and R2LTO for the different rounds of data transmission, when the initial residual of sensor nodes is the same at $100J$, is shown in Figure 4.16. The Q-value convergence rate of LACQRP approaches one faster when compared with that of RLBR and R2LTO. This is because LACQRP is centralized and has global knowledge of the network information and learns the optimal MSTs that maximize the network lifetime when taking into consideration the variable packet generation rate of the sensor nodes.

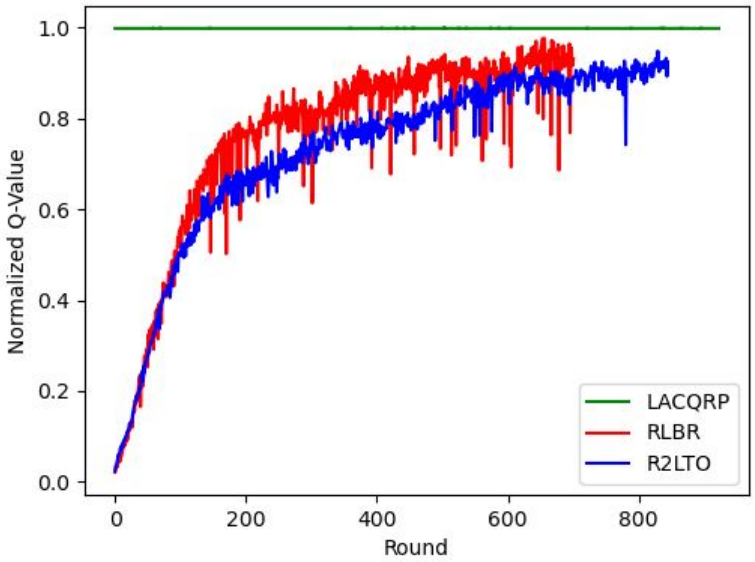

Figure 4.16: Q-Value Convergence Rate with Variable Packet Generation Rate of Sensor Node

The network lifetime of both protocols increases as the sensor nodes' residual energy increases as shown in Figure 4.17. Consequently, LACQRP has an improved network lifetime performance of 31.68% and 9.10% when compared with RLBR and R2LTO, respectively when the packet generation rate of the sensor nodes is different and the residual initial energy of the sensor nodes is increased.

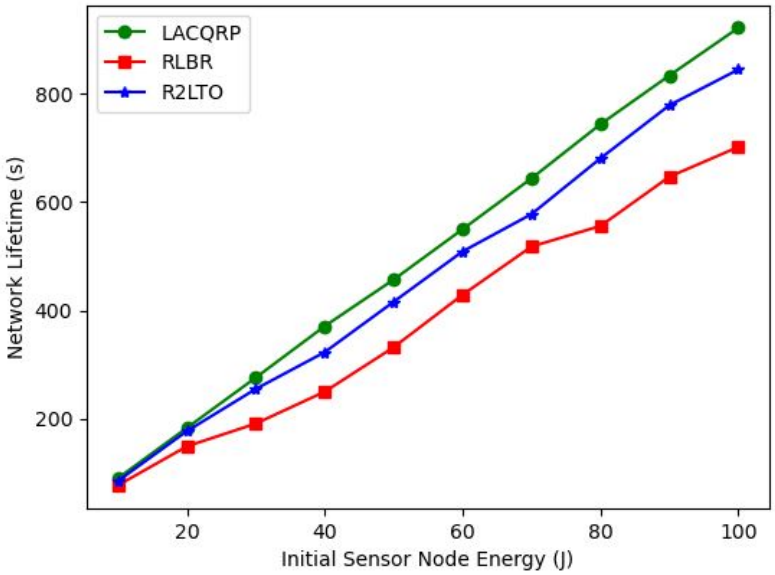

Figure 4.17: Network Lifetime with Increasing Initial Node Energy for Variable Packet Generation Rate

Likewise, the average energy consumption of the protocols increases as the initial residual energy of the sensor nodes increases as shown in Figure 4.18. Also, LACQRP has an improved average energy consumption performance of 17.39% and 26.29% when compared with RLBR and R2LTO, respectively when the packet generation rate of the sensor nodes is different and the initial residual energy of the sensor nodes is increased.

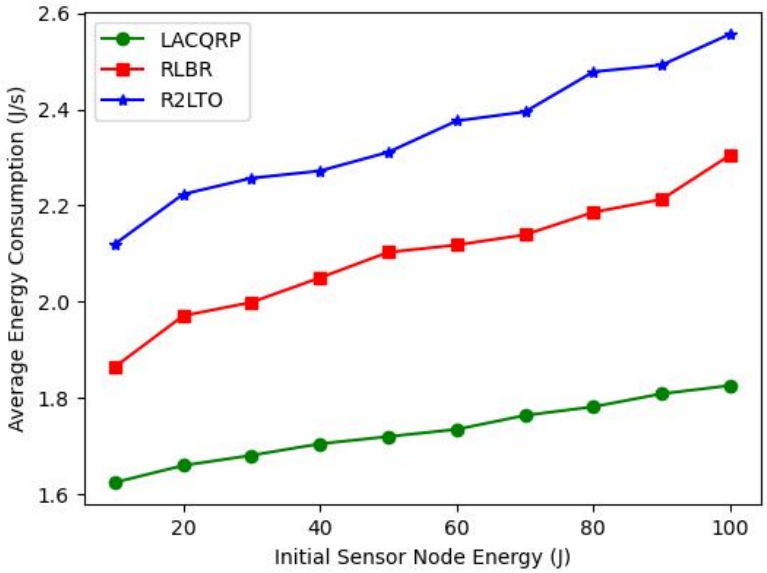

Figure 4.18: Average Energy Consumption with Increasing Initial Node Energy for Variable Packet Generation Rate

However, the performance improvement of the network lifetime and average energy consumption of LACQRP, when compared with RLBR and R2LTO when the initial residual energy of the sensor node is increasing and the packet generation rate of the sensor nodes is heterogenous, comes with a very high computation time when compared

with RLBR and R2LTO, respectively as shown in Figure 4.19. This is because of the NP-hardness of generating all MSTs by the LACQRP.

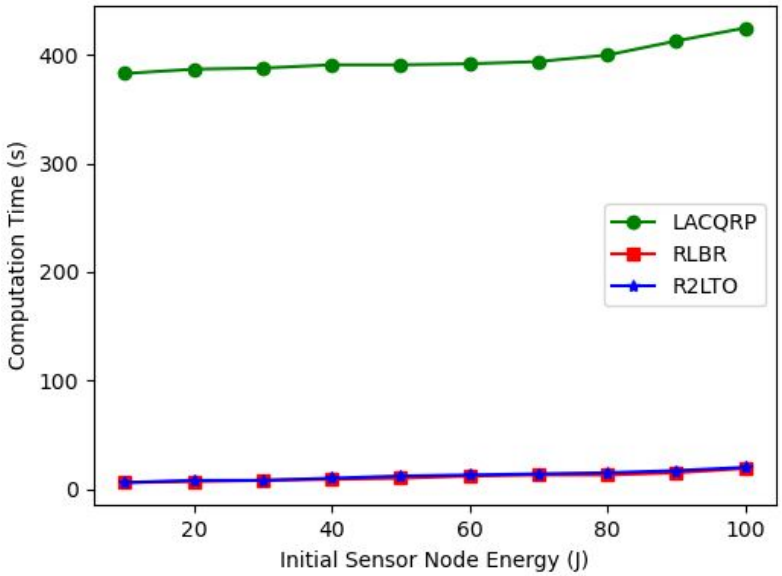

Figure 4.19: Computation Time with Increasing Initial Node Energy for Variable Packet Generation Rate

4.3.6 *Performance Comparison of LACQRP with RLBR and R2LTO for Heterogeneous IRE and PGR of Sensor Nodes*

The convergence rate, network lifetime, and average energy consumption of LACQRP are compared with RLBR and R2LTO when the sensor nodes' initial residual energy and packet generation rate are different. The sensor nodes' initial residual energy is different by assigning the energy value from the range of 10J and 100J at the multiples of 10J. Similarly, The sensor nodes' packet generation rate is different by assigning the packet generation rate value from the range of $1/s$ to $10/s$ at the multiples of $1/s$. The convergence rate of both protocols is given in Figure 4.20. The convergence rate of LACQRP approaches one faster when compared with that of RLBR and R2LTO. This is because LACQRP is centralized and has global knowledge of the network information and learns the optimal MST that maximizes the network lifetime while adapting to the heterogeneous initial residual energy and packet generation rate of the sensor nodes

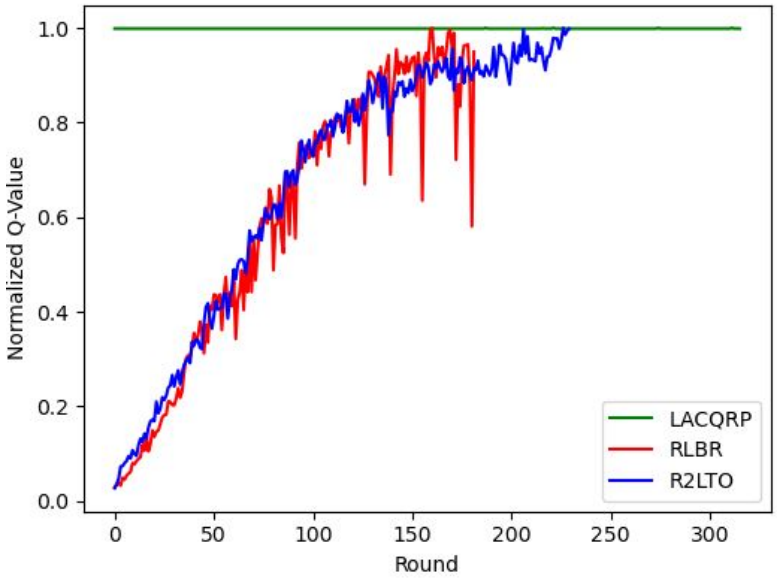

Figure 4.20: Q-Value Convergence Rate with Round of Data Transmission for Different Sensor Node PGR and IRE

This makes LACQRP has a better network lifetime performance of 68.45% and 19.96% when compared to RLBR and R2LTO, respectively for different sensor node initial residual energy and packet generation rate as shown in Figure 4.21.

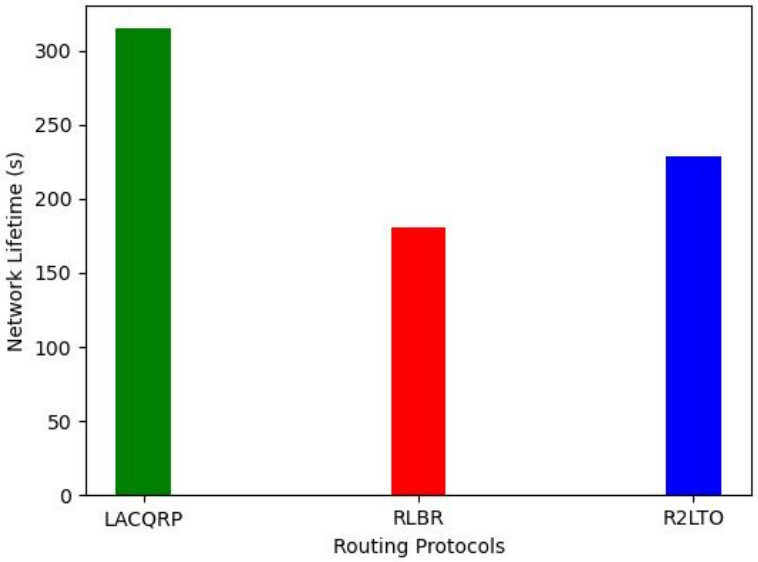

Figure 4.21: Network Lifetime for Different Sensor Node PGR and IRE

The LACQRP learns the optimal MST that minimizes the sensor nodes' energy consumption globally as shown in Figure 4.22. LACQRP has a reduced average energy consumption of 31.09% and 48.67% when compared with RLBR and R2LTO, respectively for different sensor node initial residual energy and packet generation rate.

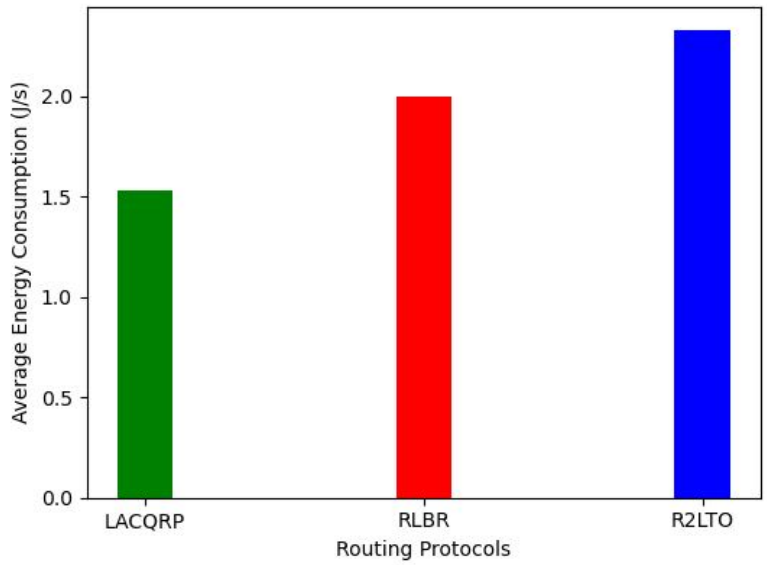

Figure 4.22: Average Energy Consumption for Different Sensor Node PGR and IRE

However, the performance improvement of the network lifetime and average energy consumption of LACQRP, when compared with RLBR and R2LTO when the initial residual energy and packet generation rate of the sensor nodes is heterogenous, comes with a very high computation time when compared with RLBR and R2LTO, respectively as shown in Figure 4.23. This is because of the NP-hardness of generating all MSTs by the LACQRP.

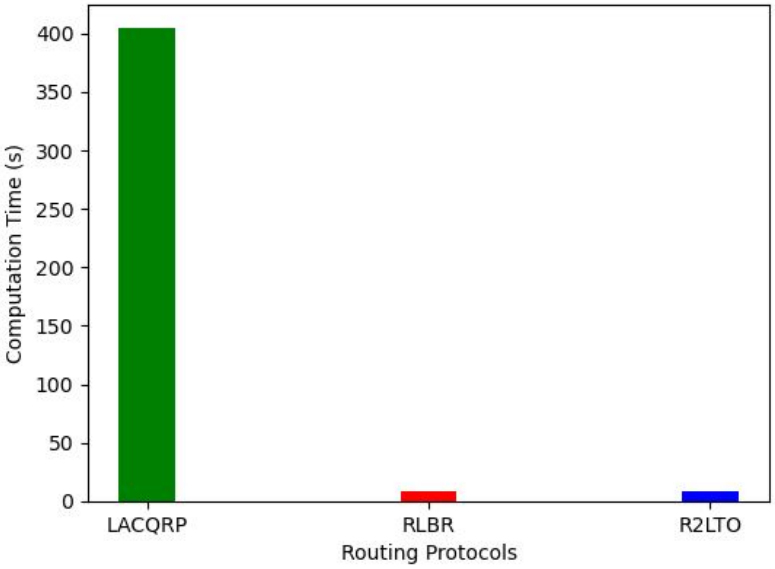

Figure 4.23: Computation Time for Different Sensor Node PGR and IRE

4.4 CONCLUSION

This chapter presents the design of a lifetime-aware centralized Q-routing protocol for WSN to maximize the network lifetime. The sink of the WSN, which also acts as the controller has global knowledge of the network information and enables the generation of

all possible distance-based MSTs which are used as routing tables. Q-learning is deployed at the controller to learn the routing path that maximizes the lifetime for the first sensor node to deplete its energy source. The proposed protocol learns the optimal MST(s) that optimizes the network lifetime and reduces the average network energy consumption for all the scenarios considered. The proposed protocol has a better convergence rate, network lifetime, and average energy consumption when compared with the distributed RL routing protocols of RLBR and R2LTO. The limitation of the proposed protocol is that it depends on an algorithm that generates all MSTs of a network graph. The problem of generating all MSTs of a graph is NP-hard (computational complexity is exponential). To remove the NP-hardness of LACQRP, the next chapter will consider a sub-optimal solution (a solution that does not guarantee that all MSTs are found, in a reasonable time). When all or a subset of MSTs are found the controller will use them to learn which ones are optimal in terms of network lifetime.

5

CENTRALIZED ROUTING FOR LIFETIME OPTIMIZATION USING GENETIC ALGORITHM AND Q-LEARNING FOR WSNS

5.1 INTRODUCTION

This chapter presents an extension of the LACQRP for WSNs. The limitation of LACQRP is the NP-hardness introduced by the All-MSTs algorithm used in generating the routing tables (RTs). Therefore a Sub-Optimal Lifetime-Aware Centralized Q-routing Protocol SOLACQRP for WSNs with GA and Q-Learning is designed to improve the time and space complexity of the LACQRP. This is achieved by replacing the All-MSTs algorithm of LACQRP with a sub-optimal solution that does not guarantee all MSTs are generated in polynomial time using a GA. The network lifetime considered is the time taken for a live sensor node not to be reachable to the sink. The sink rebuilds the network graph and runs the GA-based MSTs after the death of sensor node(s) before data transmission until the network graph is disconnected. The maximization of the network lifetime is achieved by the sink learning the MSTs at each stage of network graph building that minimizes the maximum of the sensor nodes' consumption energies using Q-Learning until the network graph is disconnected.

5.2 METHODOLOGY

The GA-based MSTs and the centralized routing for lifetime optimization Using GA and Q-Learning for WSNs are presented in the sequel.

5.2.1 *GA-based MSTs*

The set of nodes and edges of the network graph is denoted by $\mathcal{V} = \{v_1, ..., v_n\}$ and $\mathcal{E} = \{e_1, ..., e_m\} \subseteq \mathcal{V} \times \mathcal{V}$, respectively. Each edge $e \in \mathcal{E}$ is associated with an integer weight $w(e) > 0$. Denote the sum of the weights of constituent edges for an MST as weight $w(\mathcal{T})$. The problem of generating all MSTs of a graph is NP-hard [144]. That is the computational complexity is exponential. To be able to implement the LACQRP in practice, a sub-optimal solution that does not guarantee that all MSTs are found in a reasonable time is achieved using GA [142].

The distance-based MST is considered. The distance-based MST is the spanning tree having the possible minimum total cartesian distances between the connected vertices. When all or a subset of MSTs are found the sink will use them to learn which ones are optimal in terms of network lifetime. The population of the GA is obtained from MSTs generated by a classical MST algorithm [44].

The classical algorithms for finding MST of a connected undirected graph are Kruskal's algorithm [87], Prim's algorithm [113], and Boruka's algorithm [44]. Kruskal's algorithm and Boruka's algorithm can only find one MST of a network graph. This is because Kruskal's algorithm and Boruka's algorithm look at the network graph in its entirety and add the shortest edge to the existing tree until the MST is found. Subsequently, Prim's algorithm builds MST by initializing a random node as the root node. This makes Prim's

algorithm find several MSTs for a network graph that does not have distinct edge weights when varying the root node [59]. Therefore the number of MSTs generated by Prim's algorithm by varying the root node is less than or equal to the number of nodes, n of the network graph.

The Prim's algorithm as presented in Algorithm 4 is described as performing the following steps [113]:

1. Initialize a tree with a single vertex, chosen arbitrarily from the graph.
2. Grow the tree by one edge: of the edges that connect the tree to vertices not yet in the tree, find the minimum-weight edge, and transfer it to the tree.
3. Repeat step 2 (until all vertices are in the tree).

The GA-based MSTs extract unique MSTs of the network as the initial population of the GA using the classical Prim's algorithm by selecting varying nodes as the root. Prim's algorithm is called using the network graph and varying root nodes as inputs. The algorithm for generating the initial population for the genetic algorithm to find several MSTs of the network graph is given in Algorithm 11.

Algorithm 11 Generate MSTs using Prim's Algorithm

Input: $\mathcal{G}(\mathcal{V}, \mathcal{E})$
Output: $MSTs$
  $MSTs = \{\}$
  $j = 0$
  while $j < n$ **do**
    Select vertex j as the root node
    $T = Prim(\mathcal{G}, j)$ using Algorithm 4
    if $T \notin MSTs$ **then**
      $MSTs \leftarrow T$
    end if
  end while
  Return $MSTs$

The algorithm for generating the initial population MSTs using Prim's algorithm runs in $O(Nm \log n)$ time, where n, m, and N denote the number of nodes, edges, and MSTs of the network graph $\mathcal{G}$, respectively. The upper bound of N is n.

A sub-optimal solution (that is a solution that does not guarantee that all MSTs are found) in a reasonable time using a genetic algorithm is as given in Algorithm 12.

Algorithm 12 Genetic algorithm for generating MSTs

Input: cr, mr, Number of generations (NG)
Output: MSTs

$P = \{\}$
Generate $k \leq n$ unique MSTs using **Algorithm 1**
$P \leftarrow k$ unique MSTs
for $i = 1$ to NG **do**
  $P_i = \{\}$
  $n_c = \frac{100}{cr}$
  for $j = 1$ to n_c **do**
    $T_1, T_2 \in_R P$
    $\mathcal{G}_1 = T_1 \cup T_2$
    $T = Prim(\mathcal{G}_1, v)$
    if $T \notin P_i$ **then**
      $P_i \leftarrow T$
    end if
  end for
  $n_m = \frac{100}{mr}$
  for $j = 1$ to n_m **do**
    $T_3 \in_R P$
    $e_{i,j} \in_R T_3$
    $\mathcal{G}_2 = T_3 - e_{i,j}$
    $\mathcal{G}^* = \mathcal{G} - e_{i,j}$
    $Cut\ Set = \{e_{i,j} \mid e_{i,j} \in \mathcal{G}^* \ \&\ e_{i,j} \notin \mathcal{G}_2\}$
    $e^*_{i,j} \in_R Cut\ Set$
    $\mathcal{G}^*_2 = \mathcal{G}_2 + e^*_{i,j}$
    if $\mathcal{G}^*_2$ is a tree of $\mathcal{G}$ **then**
      if $\mathcal{G}^*_2 \notin P_i$ **then**
        $P_i \leftarrow \mathcal{G}^*_2$
      end if
    end if
  end for
  for T in P_i **do**
    Evaluate $fitness$ using equation 5.1
    if $fitness$ is True **then**
      if $T \notin P$ **then**
        $P \leftarrow T$
      end if
    end if
  end for
end for
Return P

Every chromosome in the population represents an MST, with its genes representing the graph edges [15]. The population evolves when passed through the crossover operator and the mutation operator, which generates new individuals (MSTs) by inheriting some of their parents' attributes (edges).

The objective function used to measure the fitness of the newly formed individual is the cost of the MST of the graph. Individual fitness is given by the possibility of the

new tree T^k formed by the crossover operator and the mutation operator having a total distance edge weight equal to the total distance edge weight of a minimum spanning tree T^* of the graph as given in Equation 5.1.

$$fitness(k) = \text{Poss}\left[\sum_{i,j \in T^k} d_{i,j} = \sum_{i,j \in T^*} d_{i,j}\right] \tag{5.1}$$

where $d_{i,j}$ for the distance-based GA MSTs is given in Equation 5.2.

$$d_{i,j} = \sqrt{(x(i) - x(j))^2 + (y(i) - y(j))^2} \tag{5.2}$$

The mutation operation is done by randomly choosing an edge, $e_{i,j}$ from a selected individual, T_3 in the population. The $e_{i,j}$ is deleted from T_3 and the main network graph $\mathcal{G}$ to form the sub-graphs, $\mathcal{G}_2$ and $\mathcal{G}^*$, respectively. A random edge, $e^*_{i,j}$ belonging to the cut set of $\mathcal{G}^*$ is added to $\mathcal{G}_2$ to form a new sub-graph, $\mathcal{G}^*_2$ [14]. The cut set of $\mathcal{G}^*$ is the set of edges belonging to $\mathcal{G}^*$ and does not belong to $\mathcal{G}_2$. The new individual generated by the mutation operation must be a tree of $\mathcal{G}$ before acceptance. The new individual generated by the mutation operator is illustrated in Figure 5.1. The mutation rate, mr is used to specify the number of times applying the mutation operator before selecting the fitness one.

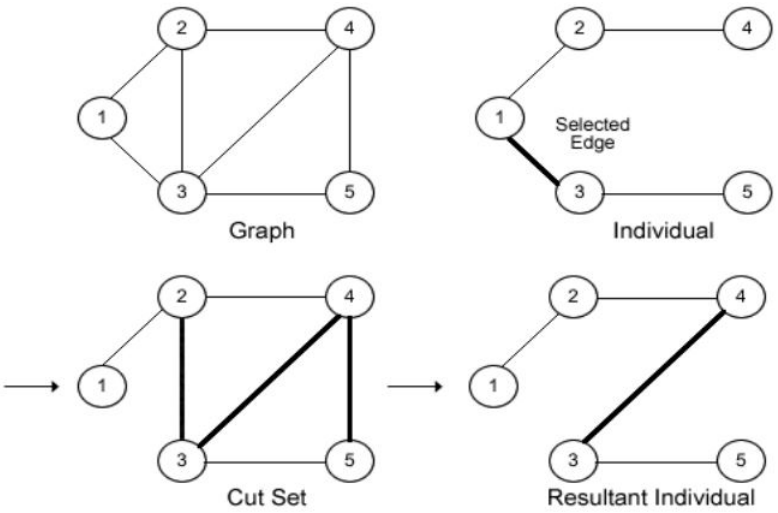

Figure 5.1: MST Mutation

The crossover operation is done by randomly selecting two individuals, T_1 and T_2 from the population as parents to form children for the next generation. T_1 and T_2 are united to form a sub-graph, $\mathcal{G}_1$ of $\mathcal{G}$ by applying the union operation [15]. The new individual generated will be an MST of $\mathcal{G}_1$ and also of the network graph $\mathcal{G}$ as illustrated in Figure 5.2. The crossover rate, cr determines how many times of applying the crossover operator before taking the fitness individual.

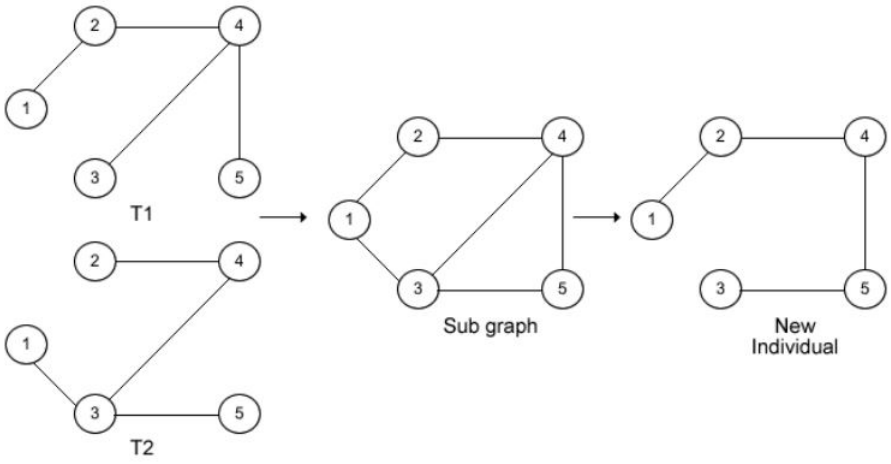

Figure 5.2: Crossover between two MSTs T1 and T2

5.2.2 *A Centralized Routing Protocol for Lifetime Optimization using GA and Q-Learning*

A Centralized Routing Protocol for Lifetime Optimization using GA and Q-Learning (CRPLOGAQL) is designed to remove the NP-hardness associated with the LACQRP. This is achieved by replacing the All-MSTs algorithm in LACQRP with the proposed GA-based MSTs. The CRPLOGAQL optimizes the time it takes for the sink not to be reachable by alive sensor node. This is achieved by finding the routing tables using Q-learning after each stage of the network graph building such that the minimum of the sensor nodes' energies is maximized. This leads to the prolonging of the time taken for sensor node(s) to die and hence the maximization of the time taken for the sink not to be reachable by alive sensor nodes.

The learning agent of the CRPLOGAQL resides in the sink of the WSN. The action space A and the state space S of the agent is the list of MSTs generated by the sink using Algorithm 12. The learning agent state is the current MST that is used by the sink in receiving data packets from the sensor nodes. The action of the learning agent is to choose an MST from the action space after a round of data transmission based on the performance of the previous MST using the predefined reward function. The learning agent measures the performance of the chosen MST by using the maximum of the energies consumption of the sensor nodes to send data packets as the reward function. This is because there is a variation in the energy consumption of the sensor nodes when different MSTs are used in data transmission. The variation in the energy consumption is from the difference in the number of links crossing each sensor node in the different MSTs.

The quality of being in a state $s \in S$ and choosing an action $a \in A$ is measured by an action-value function called Q-value. The Q-value is a measure of the long-run reward that the agent gets from each pair of state-action. The estimate of this action-value function which is used to find the best action for a given state is realized by caching the Q-values $Q(s_t, a_t)$ of pairs of state-action using the iterative update rule given in Equation 5.3. The learning of the agent is made meaningful by denoting the Q-value of the learning agent as the maximum of the sensor nodes' energy consumption when a particular MST is used in receiving data packets by the sink.

$$Q^{new}(s_t, a_t) \leftarrow (1-\alpha)Q^{old}(s_t, a_t) + \alpha\left[r_t + \gamma * \max_{a \in A}\{Q(s_{t+1}, a)\}\right] \tag{5.3}$$

The extent to which the newly learned Q-value affects the old Q-value is dependent on the learning rate, $\alpha(0,1)$. The closer the value of α is to one, the more the impact of the newly computed Q-value on the old one. If is equal to one, then the recently learned Q-value replaces the old Q-value completely. The discount factor, $\gamma[0,1]$ controls the agent's liking for the future rewards concerning the current reward. If γ is equal to 1, both the immediate reward and the future reward are considered equally.

The achievable reward r_t in each learning episode is given in Equation 5.4.

$$r_t = \max_{v \in V}\{EC_v\} \tag{5.4}$$

The EC_v is calculated by the sink after each episode using the difference between the previously estimated sensor residual energy $ESRE_v^{Previous}$ and the currently estimated sensor residual energy, $ESRE_v^{Current}$. Therefore EC_v is as given in Equation 5.5.

$$EC_v = ESRE_v^{Previous} - ESRE_v^{Current} \tag{5.5}$$

The energy model adopted for CRPLOGAQL is the same as LACQRP. The reward function is minimized by selecting the MST that has a minimum Q-value using the epsilon-greedy technique. Therefore, given a number, $r \in (0,1)$ generated randomly in each episode and a likelihood epsilon value, $\epsilon \in [0,1]$, the learning agent chooses its action in each round using the policy given in Equation 5.6.

$$a_t = \begin{cases} \underset{a \in A}{\operatorname{argmin}}\{Q_t(s,a)\}, \; if \; r < 1-\epsilon \\ Random \; action, \; otherwise. \end{cases} \tag{5.6}$$

Though no energy is consumed by the sensor nodes when there is no sending of data packets to the sink, the, $Q_0(s_0, a_0)$ is initialized as the maximum initial residual energy of the sensor nodes, E_{max}. This is to enable the fast convergence to the optimal MST(s) since the learning agent is choosing an action with the minimum Q-value and the reward of the learning agent is the maximum energy consumption of the sensor nodes.

The epsilon-greedy strategy employed by the learning agent ensures that the learning agent will converge to the optimal MST(s). The optimal MST at each stage of the network building is the MST with the highest utilization percentage. The utilization percentage of an MST is given in Equation 5.7.

$$\zeta_{MST} = \frac{\tau_{MST}}{NE} \tag{5.7}$$

where ζ_{MST} is the utilization percentage of an MST, τ_{MST} is the number of episodes the MST is used, and NE is the number of episodes before the network is rebuilt.

The proposed CRPLOGAQL for finding the optimal routing table at each stage of network building with a view to maximizing the time taken for the network graph to be disconnected is given in Algorithm 13.

Algorithm 13 CRPLOGAQL

Input: $G(V, E)$, α, γ, ϵ, Learning round (L)
Output: Optimal MST(s)

```
Controller executes Algorithm 12
for i = 1 to L do
    Initialize Q_0(s_0, a_0) = E_max.
    Initialize s_0 as a random MST
    Select an MST using Equation 5.6.
    Broadcast the MST to all sensor nodes.
    Sink receives data from the sensors using the MST.
    Computes the reward using Equation 5.4.
    Updates Q^new using Equation 5.3.
    Updates s_t as the current MST.
    if Any sensor dies, then
        Delete the sensor(s) from G(V, E)
        Delete links connected to the dead sensor(s)
        Rebuild G(V, E)
        if G(V, E) is connected, then
            Do step 1
            Do steps 3 to 14
        else
            break
        end if
    end if
end for
```

The initialized Q-matrix of Algorithm 13 depends on the number of generated MSTs, N by Algorithm 12 that are used as routing tables. This makes the time complexity of Algorithm 13 to be the same as Algorithm 12. Likewise, Algorithm 13 has a space complexity of $O(N)$.

5.3 SIMULATION AND RESULTS DISCUSSIONS

The performance of the proposed GA-based MSTs is first established for convergence using simulations and compared with the All-MSTs algorithm [144] using the number of MSTs generated and computation time as the performance indices. Subsequently, the performance of the CRPLOGAQL is achieved by simulations using the performance metrics of network lifetime, energy consumption, number of alive sensor nodes (NAN), and computation time. These metrics of the CRPLOGAQL are compared with that of the LACQRP as a means of validation. The network lifetime is computed as the time taken for the sink not to be reachable by alive sensor node(s). The NAN is the number of alive sensor nodes in the network lifetime. The computation time is the central processing unit (CPU) time to achieve the network lifetime. The CRPLOGAQL and LACQRP are coded with Python 3.8 under the "PyCharm" development environment. The graphical structure of the WSN is implemented using the python networkx module [57]. The Python code is executed on the SLURM (Simple Linux Utility for Resource Management) cluster on the IRIT's OSIRIM platform. The Computer nodes of the OSIRIM platform adopted are the same as that of Section 3.3.

5.3.1 *Performance Comparison of GA-based MSTs with All-MSTs Algorithm*

The choice of the crossover rate and mutation rate is achieved by combining the crossover rate and mutation rate that yield the best convergence to a good number of MSTs in a reasonable time for the deployed network which is the same as in Figure 3.1.

Figure 5.3 shows the number of MSTs generated by the GA-based MSTs when the mutation rate is kept constant at 1 and the crossover rate is chosen among 0.01, 0.1, and 1.

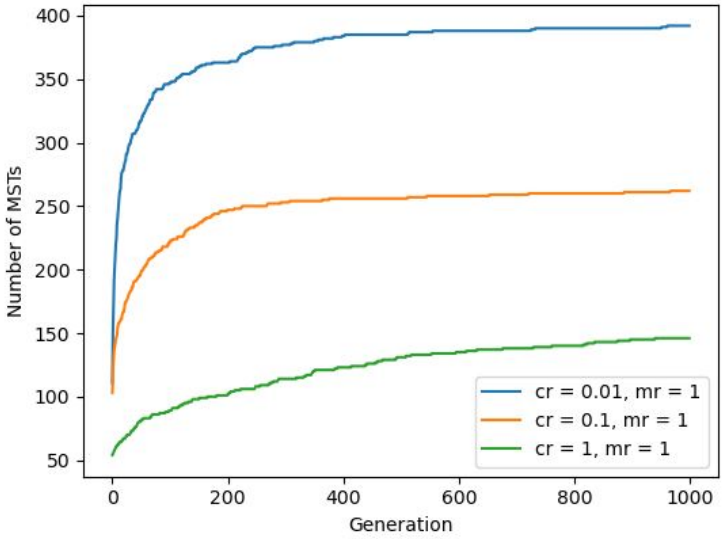

Figure 5.3: Number of MSTs in each Generation with varied Crossover Rates.

When the crossover is 0.01 the genetic algorithm for generating MSTs converges to a higher number of MSTs which is 376 for 1000 generations. This is because the lower the crossover rate, the higher the number of crossover operations that are performed by the GA-based MSTs. This leads to a higher possibility of finding newer individuals in a generation. A crossover rate of 0.01 means 100 crossover operations in a given generation. Therefore, the lower the crossover rate, the higher the speed of convergence to a larger number of MSTs. There is a direct relationship between the speed of convergence and the crossover rate of the GA-based MSTs because the crossover operation always leads to the formation of an MST of the network graph.

The performance of the GA-based MST for constant mutation rate and varying crossover rate when compared with the All-MSTs algorithm using the number of MSTs generated and the computation time is given in Table 5.1.

Table 5.1: Performance Comparison of GA-based MSTs with All-MSTs algorithm for varying Crossover Rate and fixed Mutation Rate.

GA-based MSTs			**All-MSTs**	
Crossover Rate	**MSTs**	**Time (s)**	**MSTs**	**Time (s)**
0.01	376	47.28	4723	391.29
0.1	262	21.53		
1	146	12.78		

As seen in Table 5.1, the GA-based MSTs finds the highest number of MSTs (376) for 1000 generations when the crossover rate is 0.01 and the mutation rate is kept constant at 1. When compared with the All-MSTs algorithm, this resulted in 7.96% of the number of the MSTs generated by the All-MSTs algorithm with a reduced computation time of 87.97%.

Consequently, to examine the effect of the mutation rate on the GA-based MSTs, Figure 5.4 shows the number of MSTs generated by the GA-based MSTs when the crossover rate is kept constant at 1 and the mutation rate is chosen among 0.01, 0.1, and 1.

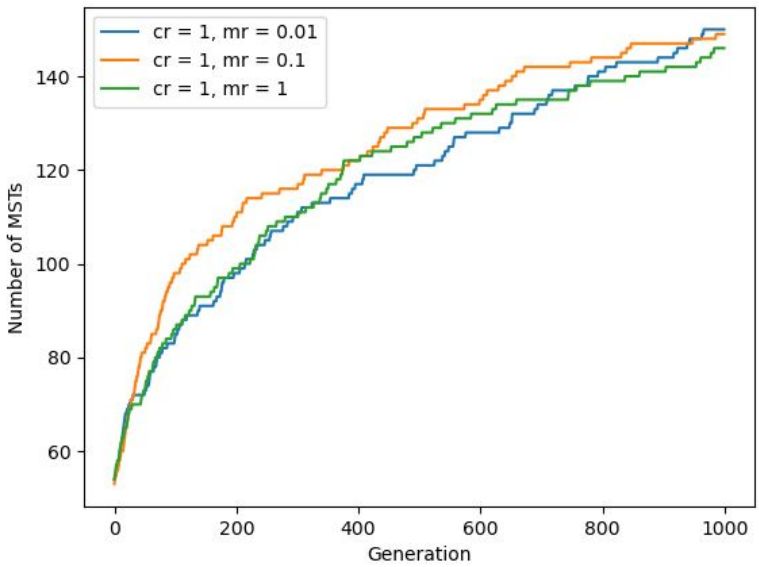

Figure 5.4: Number of MSTs in each Generation with varied Mutation Rates.

When the crossover rate is kept constant and the mutation rate is increased, the number of mutation operations that are performed by GA-based MSTs is reduced. This leads to a reduced possibility of finding newer MSTs in a given generation. Though there is no direct relationship between the speed of convergence and the mutation rate of the GA-based MSTs. This is because the mutation operation does not always lead to the formation of an MST of the network graph. The performance of the GA-based MST for constant crossover rate and varying mutation rate when compared with the All-MSTs algorithm using the number of MSTs generated and the computation time is given in Table 5.2.

Table 5.2: Performance Comparison of GA-based MSTs with All-MSTs Algorithm for varying Mutation Rate and fixed Crossover Rate.

GA-based MSTs			**All-MSTs**	
Mutation Rate	**MSTs**	**Time (s)**	**MSTs**	**Time (s)**
0.01	150	327.62	4723	391.29
0.1	149	50.87		
1	146	12.78		

As seen in Table 5.2, the lower the mutation rate, the higher the computation time. This is because the lower the mutation rate the higher the number of mutation operations performed in a given generation. However, this higher number of mutation operations attributed to the lower mutation rate does not yield a significantly higher number of new MSTs generated in a given generation when compared to a higher mutation rate. This is because the mutation operation does not always lead to the formation of an MST as against the crossover operator that always produces an MST of the deployed network. Therefore, the crossover rate, mutation rate, and the number of generations of the GA-based MSTs used to implement the CRPLOGAQL are justified from Table 5.1 and Table 5.2, respectively, and are given in Table 5.3. The learning rate, discount factor and epsilon used to implement the Q-Learning in CRPLOGAQL are the same as that of LACQRP.

Table 5.3: Simulation Parameters of GA-Based MSTs for CRPLOGAQL

Parameters	**Values**
Crossover rate	0.01
Mutation rate	1
Number of generations	1000

5.3.2 *Performance Comparison of CRPLOGAQL with LACQRP for Homogeneous Sensor Nodes IRE and PGR*

Figure 5.5 shows the comparison of the number of alive nodes in each round of data transmission of CRPLOGARL with that of the LACQRP when the initial residual energy and packet generation rate of the sensor nodes are kept arbitrarily at 10 J and 1 $/s$, respectively.

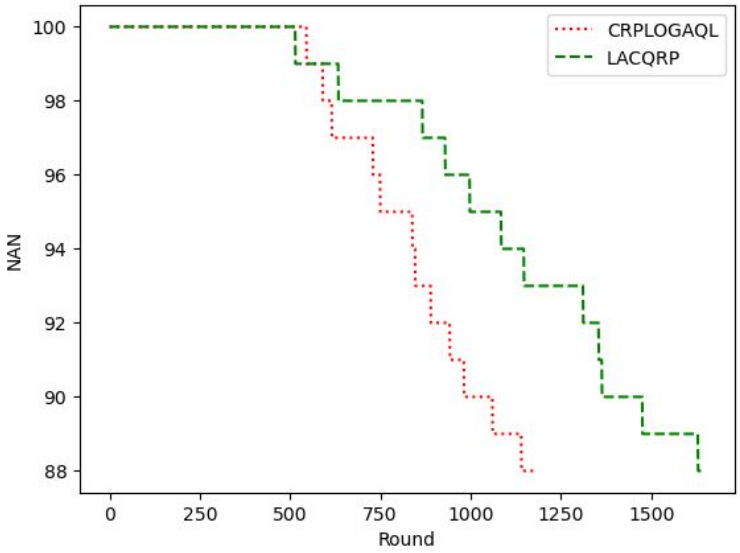

Figure 5.5: Number of alive Sensor Nodes with Data Transmission Round.

The number of alive sensor nodes of both routing protocols decreases with the network lifetime. The decrease in the number of alive sensor nodes is due to the depletion of the energy sources of the sensor nodes. As seen in Figure 5.5, mostly the number of

rounds of data transmission for CRPLOGARL between the death of the sensor nodes is lower than that of LACQRP. Therefore the CRPLOGAQL makes the network graph to be disconnected faster when compared with LACQRP. This is because CRPLOGAQL only uses a subset of all MSTs that does not contain all the optimal ones. This led to a reduced network lifetime in CRPLOGAQL when compared with LACQRP.

Subsequently, the network lifetime with increasing the sensor nodes' initial residual energies from 1 J to 10 J for both routing protocols and the packet generation rate of the sensor node is kept at 1 $/s$ arbitrary is as shown in Figure 5.6.

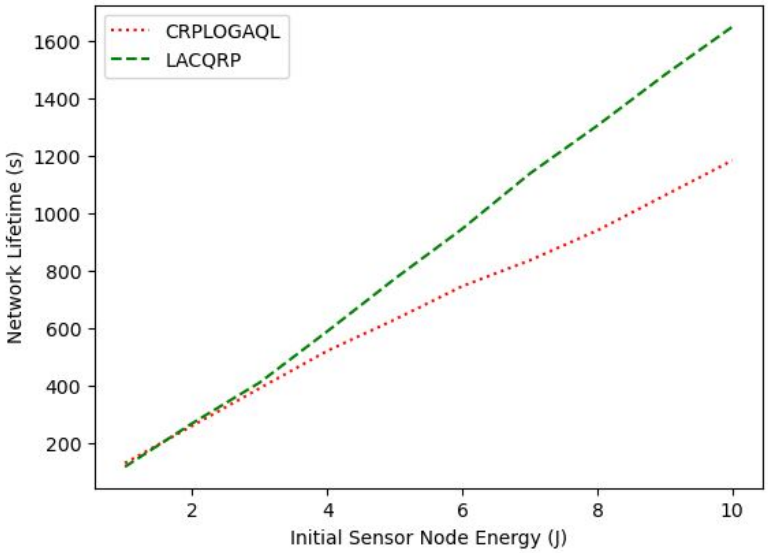

Figure 5.6: Network Lifetime with Initial Sensor Node Energy.

As the initial sensor nodes' residual energies increase, the network lifetime of both routing protocols increases. This is because the network lifetime is proportional to the residual energies of the sensor nodes. At lower initial residual energy of the sensor nodes, that is from 1 J to 3 J both protocols tend to have the same network lifetime, this is because LACQRP utilizes all MSTs which is large in size compared to the MSTs generated by the GA-based MSTs. Therefore, at the smaller initial residual energy of the sensor nodes, LACQRP has not fully learned the optimal MSTs before the network lifetime. But as the initial residual energy of the sensor nodes increases considerably, the network lifetime of CRPLOGAQL is lower than LACQRP, this is because LACQRP is using the optimal MSTs to send packets to the sink and CRPLOGAQL only uses a subset of all MSTs of the network graph which does not contain all the optimal ones to maximize the network lifetime. The use of the subset of all MSTs by the CRPLOGAQL also makes the protocol consume more energy as the initial residual energy of the sensor nodes increases when compared with the LACQRP as shown in Figure 5.7. This is attributed to the non-use of the optimal MSTs in sending the packet to the sink as compared with the LACQRP which utilizes all possible MSTs of the network graph as the state space and action space.

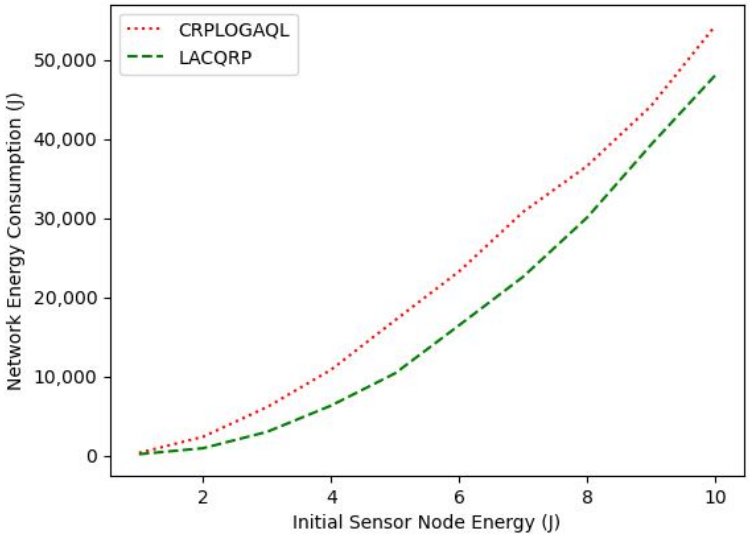

Figure 5.7: Network Energy Consumption with Initial Sensor Node Energy.

The CRPLOGAQL has a lower network lifetime performance of 22.70 % and higher network energy consumption of 27.18 % when compared with LACQRP for fixed packet generation rate and increasing initial residual energy of the sensor nodes. However, due to the NP-hardness of generating all MSTs in LACQRP, CRPLOGARL has a reduced computation time of 95.26 % when compared with LACQRP as shown in Figure 5.8.

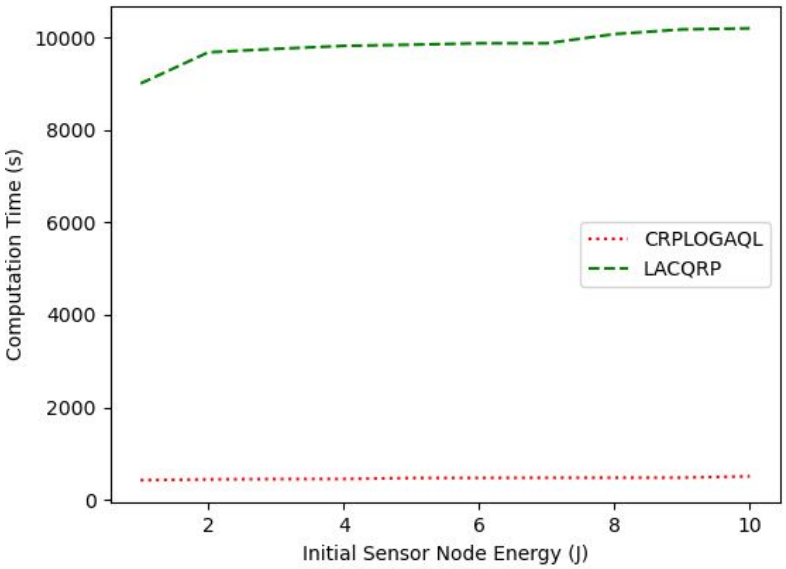

Figure 5.8: Computation Time with Initial Sensor Node Energy.

Subsequently, the network lifetime, network energy consumption, and computation time performance of the CRPLOGAQL are compared with LACQRP for constant initial residual energy and increasing packet generation rate of the sensor nodes. The initial residual energy of the sensor nodes is kept constant at $10J$ while varying the packet generation rate from $1/s$ to $10/s$. Figure 5.9 shows the network lifetime for increasing sensor nodes' packet generation rate for both routing protocols.

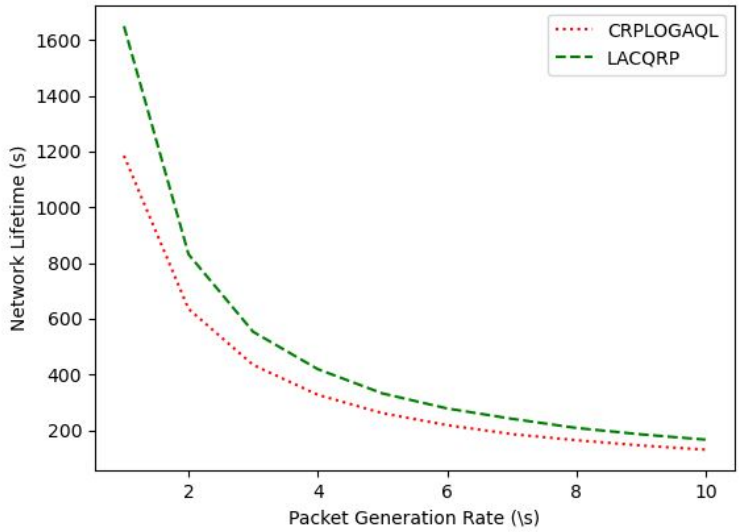

Figure 5.9: Network Lifetime with Packet Generation Rate of Sensor Nodes.

As the packet generation rate increases, the network lifetime of both routing protocols decreases. This is because an increase in packet generation rates of the sensor nodes will result in to increase in the energy consumption of the sensor. This causes the faster depletion of the energy of the sensor nodes and regeneration of the network graph by the sink until the network is disconnected and alive sensor nodes cannot forward packets to the sink since the is no path to reach the sink.

The increase in the packet generation rate of both protocols resulted in a decrease in the network energy consumption for both protocols as shown in Figure 5.10.

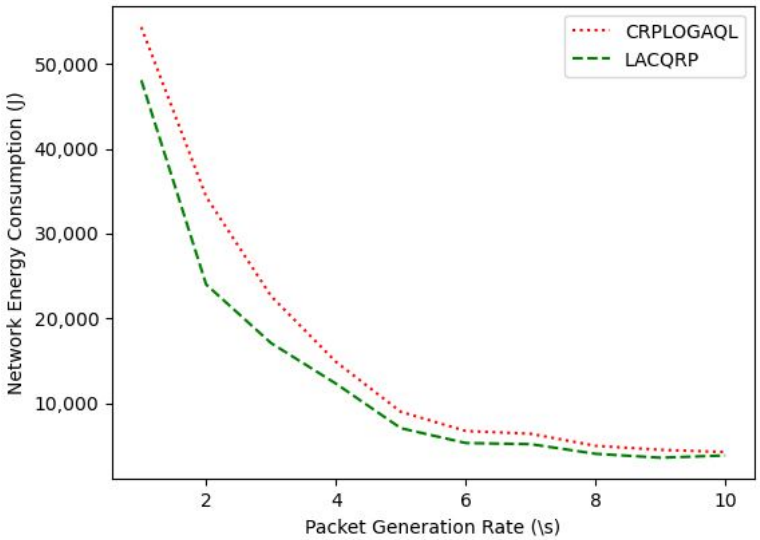

Figure 5.10: Network Energy Consumption with Initial Sensor Node Energy.

This is because of an increase in the number of data packets generated and transmitted by the sensor nodes and the rounds of data transmission it takes for the sensor nodes to deplete their energy decreases. Since the network energy consumption is the difference between the summation of the residual initial energy before any data transmission and the summation of the remaining energy of the sensor nodes after the network is

disconnected, as the network lifetime decreases as the packet generation rate increases, the network energy consumption also decreases.

The CRPLOGAQL has a lower network lifetime performance of 24.22 % and higher network energy consumption of 24.20 % when compared with LACQRP for increasing packet generation rate and fixed initial residual energy of the sensor nodes. However, due to the NP-hardness of generating all MSTs in LACQRP, CRPLOGARL has a reduced computation time of 95.05 % when compared with LACQRP as shown in Figure 5.11.

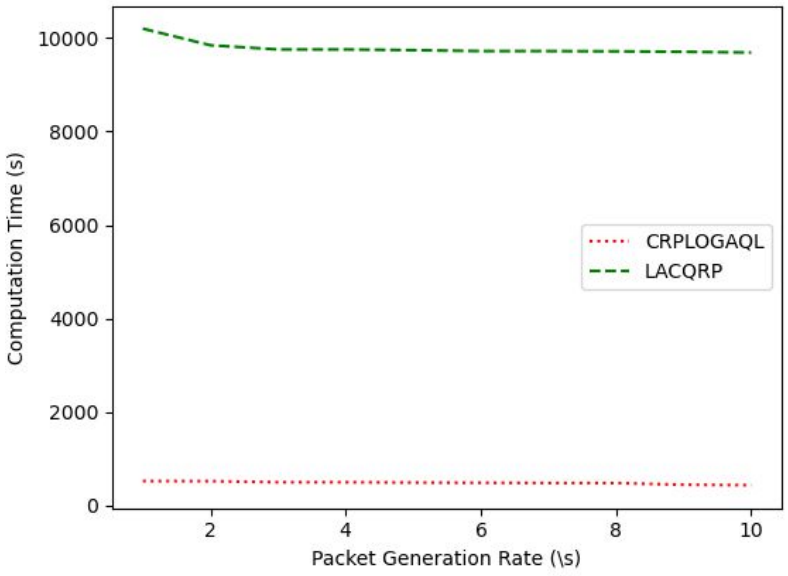

Figure 5.11: Computation Time with increasing PGR and fixed IRE of Sensor Nodes.

5.3.3 *Performance Comparison of CRPLOGAQL with LACQRP for Heterogeneous PGR and Homogeonous IRE of Sensor Nodes*

In a more practical scenario, the packet generation rate of sensor nodes is heterogeneous. Therefore the performance of CRPLOGAQL is compared with that of LACQRP for heterogeneous packet generation rate and homogenous initial residual energy of sensor nodes. All sensor node generates and transmit a different number of packet(s) (between 1 and 10 inclusive) in every round of the data transmission until the network is disconnected. To accommodate the large number of packets in the network and the number of rounds required by both protocols to learn the best paths to maximize the network lifetime, the initial residual energy of the sensor nodes is increased arbitrarily from 10 J to 100 J. This is because the centralized agent mostly for LACQRP requires a long term of learning to find the optimal paths due to the state space and action space.

Figure 5.12 shows the comparison of the number of alive nodes in each round of data transmission of CRPLOGARL with that of the LACQRP when the initial residual energy of the sensor nodes is set constant arbitrarily at 100 J and the packet generation rate of the sensor nodes are heterogenous as stated.

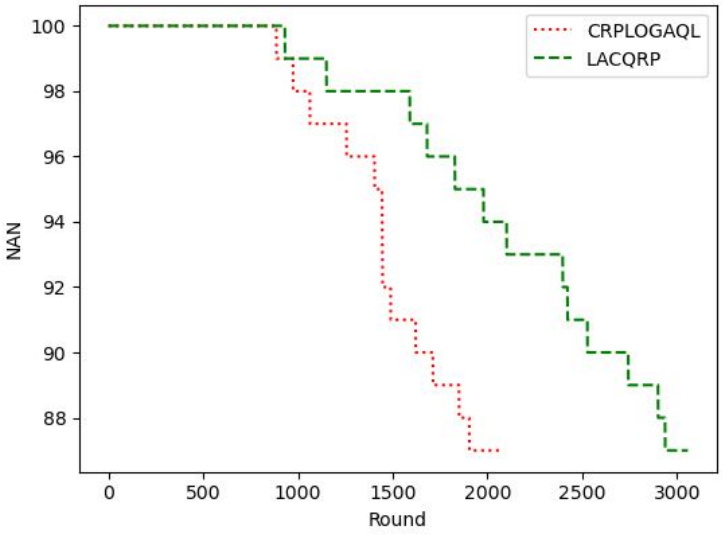

Figure 5.12: NAN with Data Transmission Round for Heterogeneous Sensor Nodes PGR

The number of alive sensor nodes of both routing protocols decreases as the round of data transmission increases for the scenario of heterogeneous traffic. The decrease in the number of alive sensor nodes is due to the depletion of the energy sources of the sensor nodes as the round of data transmission increases. As seen in Figure 5.12, mostly the number of rounds of data transmission for CRPLOGARL between the depletion of the energy of the sensor nodes is lower than that of LACQRP for the heterogeneous traffic at the sensor nodes. Therefore the CRPLOGAQL makes the network graph to be disconnected faster when compared with LACQRP for heterogeneous traffic. This is because CRPLOGAQL only uses a subset of all MSTs that does not contain all the optimal ones. This led to a reduced network lifetime in CRPLOGAQL when compared with LACQRP. The optimal MSTs give the optimal routes to transmit the different number of packets generated by each sensor node to the sink in a way that the sensor nodes energy consumption is balanced as the round of data transmission increases, minimizing the energy consumption by each sensor node, and prolonging the network lifetime. Figure 5.13 shows the network lifetime of CRPLOGAQL and LACQRP for heterogeneous traffic at the sensor nodes when the initial residual energy is homogenous and increased from $10J$ to $100J$.

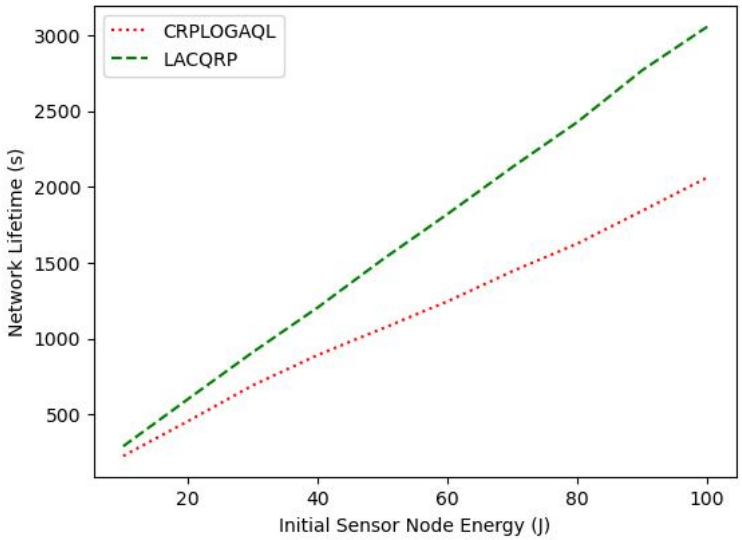

Figure 5.13: Network Lifetime with increasing Initial Sensor Node Energy for Heterogeneous Sensor Nodes PGR

As seen in Figure 5.13 as the initial residual energy of the sensor nodes increases with the different traffic, the network lifetime of both protocols also increases. The higher the initial residual energy of the sensor nodes, the more round of data transmission it takes each sensor node to deplete its energy source. Subsequently, as the initial residual energy of the sensor nodes increases, LACQRP performs better than the CRPLOGAQL in terms of network lifetime and network energy consumption. The higher the initial residual energy of the sensor nodes, the more the number of rounds the sink in CRPLOGAQL utilizes the non-optimal MSTs to receive the different packets from the sensor nodes. This causes CRPLOGAQL to consume more energy as the initial residual energy of the sensor nodes increases as shown in Figure 5.14.

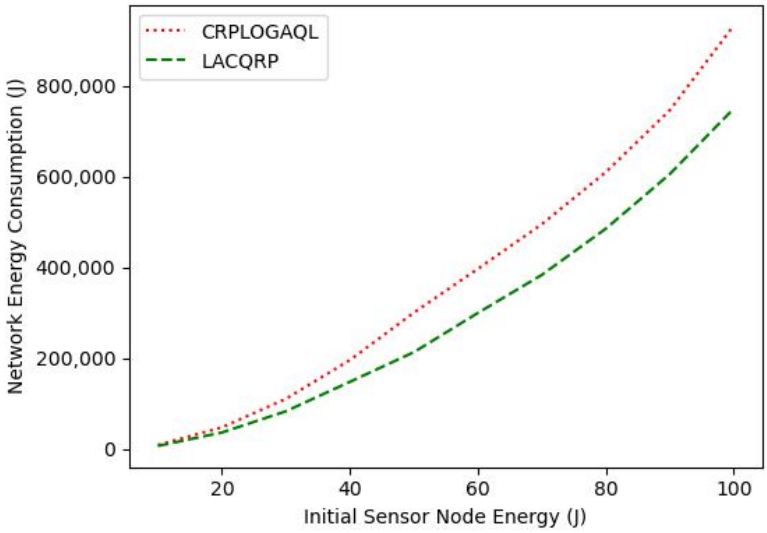

Figure 5.14: Network Energy Consumption with increasing Initial Sensor Node Energy for Heterogeneous Sensor Nodes PGR

The CRPLOGAQL has a lower network lifetime performance of 30.10% and higher network energy consumption of 21.70% when compared with LACQRP for increasing

homogenous initial residual energy and different packet generation rate of the sensor nodes. However, due to the NP-hardness of generating all MSTs in LACQRP, CRPLOGARL has a reduced computation time of 90.19% when compared with LACQRP as shown in Figure 5.15.

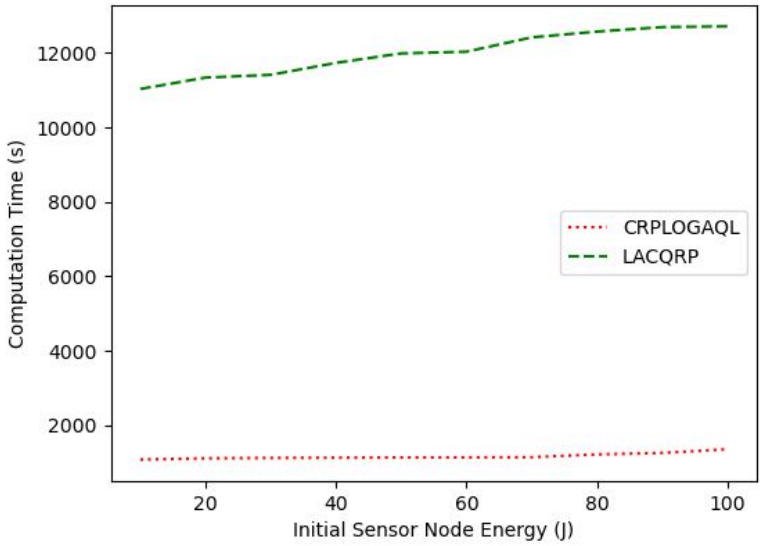

Figure 5.15: Computation Time with increasing Initial Sensor Node Energy for Heterogeneous Sensor Nodes PGR

5.3.4 *Performance Comparison of CRPLOGAQL with LACQRP for Heterogeneous IRE and Homogeonous PGR of Sensor Nodes*

The NAN, network lifetime, network energy consumption, and computation time performance of CRPLOGAQL is compared with that of LACQRP for heterogeneous initial residual energy and homogenous packet generation rate of sensor nodes. Each sensor node has different initial residual energy that is randomly chosen between $10J$ and $100J$ inclusive by the multiple of $10J$ while the packet generation rate of the sensor nodes is fixed and arbitrarily increased from $1/s$ to $10/s$. Figure 5.16 shows the comparison of the NAN in each round of data transmission of CRPLOGARL with that of the LACQRP when the packet generation rate of the sensor nodes is set constant arbitrarily at $1/s$ and the initial residual energy of the sensor nodes are heterogeneous. The network lifetime and NAN performance of CRPLOGAQL are degraded when compared with LACQRP when the initial residual energy of the sensor nodes is heterogeneous. This is attributed to the subset of all MSTs of the network utilized by the CRPLOGAQL. Therefore CRPLOGAQL does not learn all the optimal MSTs used as routing paths by the sensor nodes to maximize the network lifetime for heterogeneous initial residual energy of the sensor nodes. This makes the round of data transmission on average between the reconstruction of the network graph by the sink while using CRPLOGAQL to be lower when compared with LACQRP.

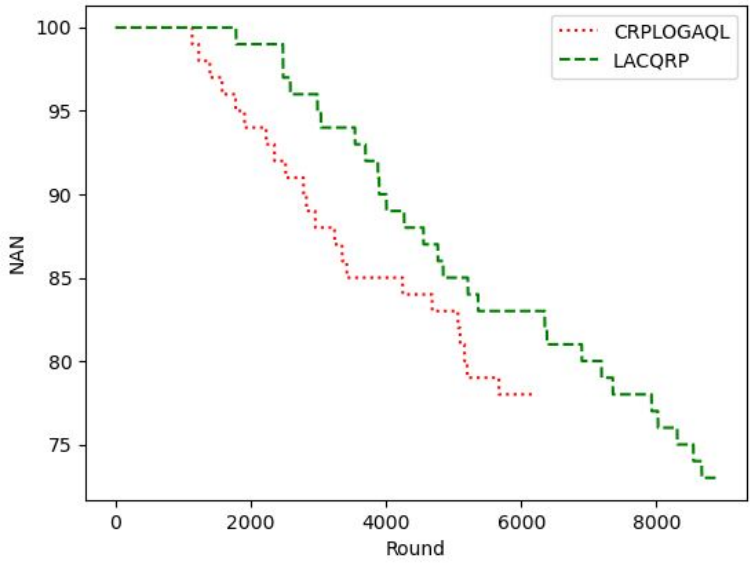

Figure 5.16: NAN with Data Transmission Round for Heterogeneous Sensor Nodes Initial Energy

The CRPLOGARL has 78 alive sensor nodes when the network graph is disconnected. This is against the LACQRP with 73 alive sensor nodes before the sink is no longer reachable for data transmission. This resulted in CRPLOGAQL having 6.85% degradation in NAN performance compared to LACQRP. Therefore the CRPLOGARL makes the network graph to be disconnected faster when compared with LACQRP. Figure 5.17 shows the network lifetime of CRPLOGAQL and LACQRP for heterogeneous initial residual energy at the sensor nodes when the packet generation rate is homogenous and increases from $1/s$ to $10/s$.

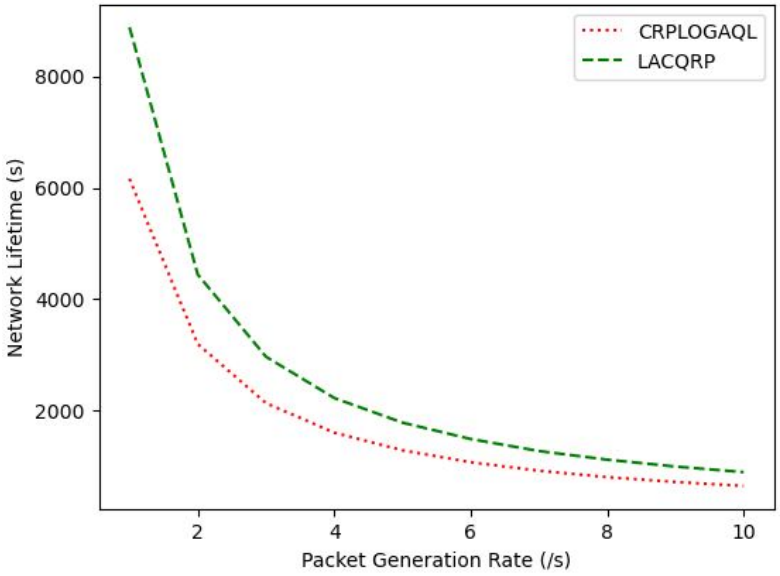

Figure 5.17: Network Lifetime with increasing Sensor Node PGR for Heterogeneous Sensor Nodes Initial Energy

The CRPLOGAQL has a lower network lifetime performance of 28.97% and higher network energy consumption of 40.12% when compared with LACQRP for increasing

homogenous packet generation rate and heterogenous initial residual energy of the sensor nodes as shown in Figure 5.17 and Figure 5.18, respectively.

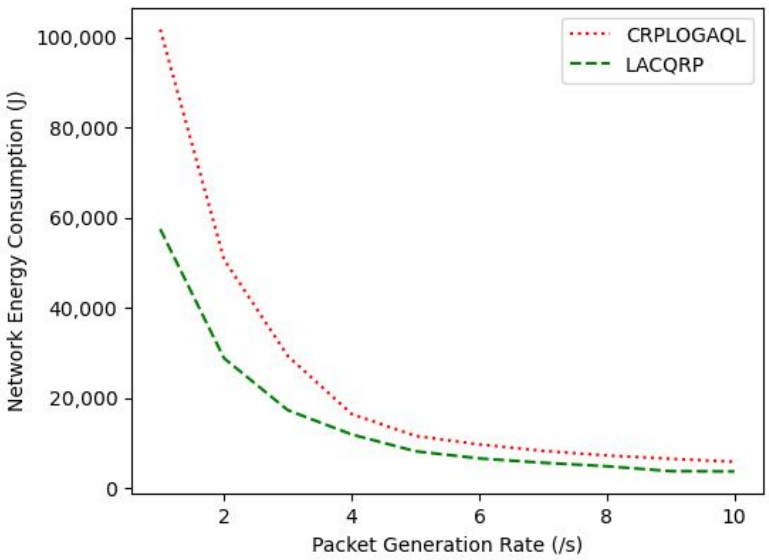

Figure 5.18: Network Energy Consumption with increasing Sensor Node PGR for Heterogeneous Sensor Nodes Initial Energy

However, due to the NP-hardness of generating all MSTs in LACQRP, CRPLOGARL has a reduced computation time of 87.10% when compared with LACQRP as shown in Figure 5.19.

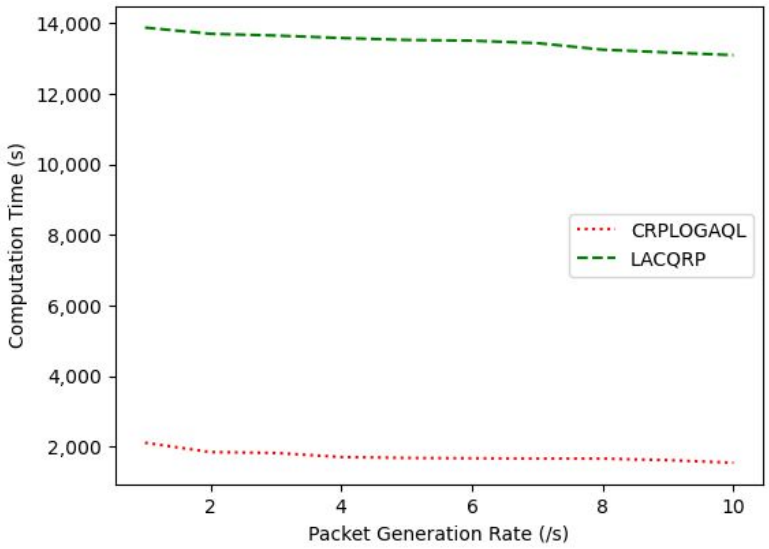

Figure 5.19: Computation Time with increasing Sensor Node PGR for Heterogeneous Sensor Nodes Initial Energy

5.3.5 *Performance Comparison of CRPLOGAQL with LACQRP for Heterogeneous PGR and IRE of Sensor Nodes*

The NAN, network lifetime, network energy consumption, and computation time performance of CRPLOGAQL is compared with that of LACQRP when the sensor nodes IRE and PGR are heterogeneous. This is achieved by choosing the IRE of the sensor nodes randomly among 10J and 100J inclusive of the multiples of 10J. Also, the PGR of each sensor node is randomly chosen among 1/s and 10/s inclusive of the multiples of 1/s. Figure 5.20 and Figure 5.21 show the NAN and network lifetime performance of CRPLOGAQL when compared with LACQRP, respectively.

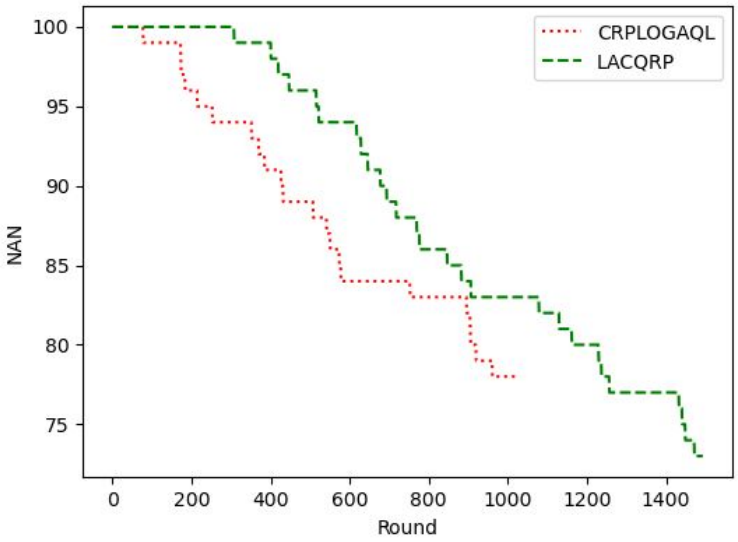

Figure 5.20: NAN with Data Transmission Round for Heterogeneous Sensor Nodes IRE and PGR

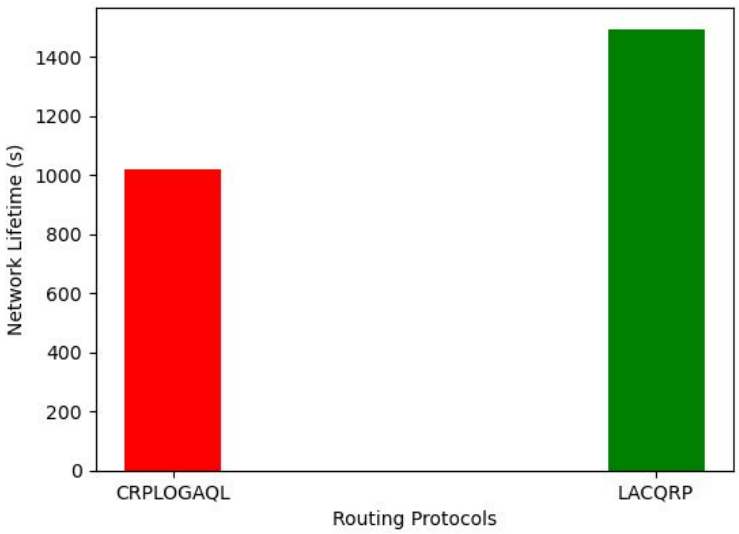

Figure 5.21: Network Lifetime for Heterogeneous Sensor Nodes IRE and PGR

As shown in Figure 5.20, the network disconnected faster with CRPLOGAQL when compared with LACQRP. Also, the CRPLOGARL has 78 alive sensor nodes when

the network graph is disconnected. This is against the LACQRP with 73 alive sensor nodes before the sink is no longer reachable for data transmission. This resulted in CRPLOGAQL having 6.85% and 31.66% degradation in NAN and network lifetime, respectively performance compared to LACQRP. This is because the MST utilized by the CRPLOGAQL in most of the rounds of data transmission is not optimal to maximize the network lifetime for the scenario of heterogeneous traffic and IRE of sensor nodes. This causes CRPLOGAQL to consume 29.97% more network energy with respect to LACQRP as shown in Figure 5.22.

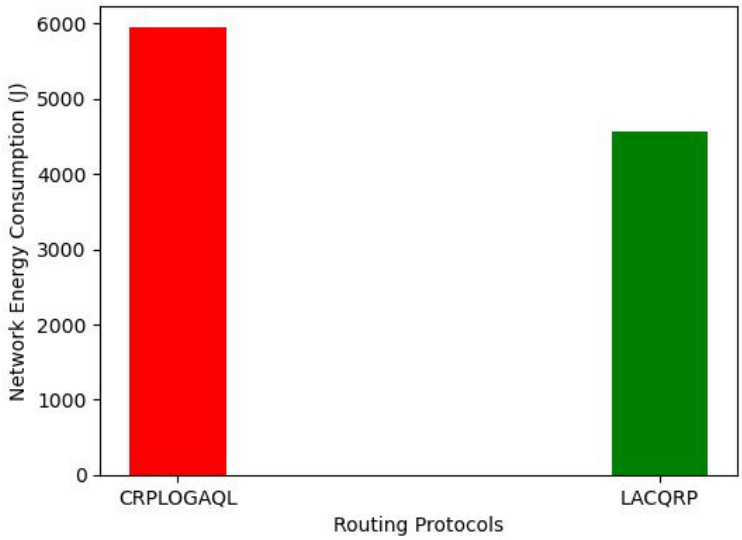

Figure 5.22: Network Energy Consumption for Heterogeneous Sensor Nodes IRE and PGR

However, the degradation of the NAN, network lifetime, and network energy consumption performance of CRPLOGAQL to LACQRP comes with an improvement in the computation time of 89.45% as shown in Figure 5.23.

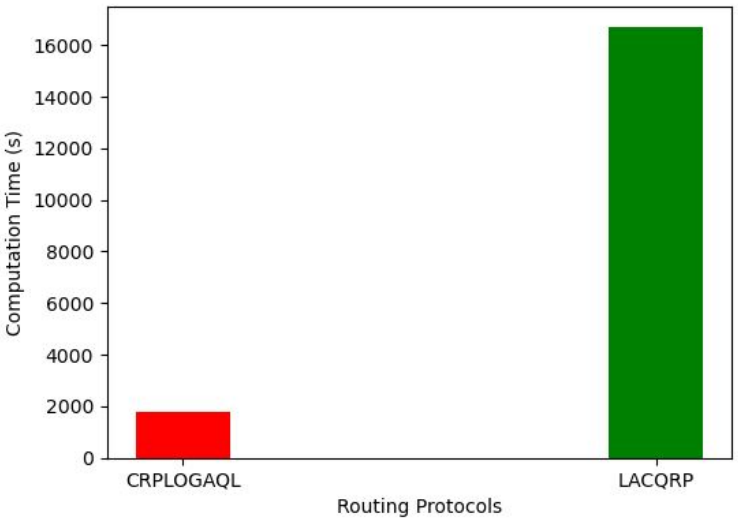

Figure 5.23: Computation Time for Heterogeneous Sensor Nodes IRE and PGR

5.4 CONCLUSION

This chapter presented the design of a centralized routing protocol for lifetime optimization using GA-based MSTs and Q-Learning for WSNs. The sink which contains the learning agent generated subsets of all minimum spanning trees of the network graph in polynomial time for routing purposes using the designed GA-based MSTs. The Q-Learning deployed at the sink learned the optimal MST(s) for lifetime optimization at each stage of the network building until the sink is not reachable by alive sensor nodes. This maximized the time taken for the sink not to be reachable by alive sensor nodes. The centralized routing protocol for lifetime optimization using GA-based MSTs and Q-Learning has improved computation time when compared with the lifetime-aware centralized Q-routing protocol that uses all minimum spanning trees of the network graph. This can enable the real-life implementation of the centralized routing protocol for lifetime optimization using GA-based MSTs and Q-Learning for WSNs feasible. However, the centralized routing protocol for lifetime optimization using GA-based MSTs and Q-Learning suffered a reduction in the performance of the network lifetime, network energy consumption, and the number of alive nodes when the network graph is disconnected. This is because the subset of the MSTs generated by the GA-based MSTs does not contain all the optimal MST(s) and Q-Learning requires a large learning round to find the few optimal MSTs. The next chapter will consider replacing Q-Learning with LSPI so as to increase lifetime and energy consumption performance.

6

CENTRALIZED ROUTING FOR LIFETIME AND ENERGY OPTIMIZATION IN WSNS USING GENETIC ALGORITHM AND LEAST-SQUARE POLICY-ITERATION

6.1 INTRODUCTION

The Centralized Routing Protocol for Lifetime Optimization using Genetic Algorithm and Q-Learning (CRPLOGAQL) presented in Chapter 5, use Q-Learning to find the optimal routing path. However, due to the large state space and action space of the protocol, the baseline Q-Learning used to implement the protocol suffers from degradation in the network energy consumption and network lifetime due to the large number of learning episodes required to learn the optimal routing path. Also, Q-Learning is very sensitive to parameter settings, for example, changes in the learning rate affect the network lifetime. To overcome these limitations, a highly efficient model-free RL technique called Least-Square Policy Iteration (LSPI) is used to replace Q-Learning. This is because LSPI chooses a routing path in a given state considering all possible routing paths and it's not sensitive to the learning rate. Also, while Q-learning evaluates the optimal policy from the Q-values, LSPI updates the Q-values based on the most updated information regarding the network dynamics using weighted functions. Therefore, this chapter presents the design of a Centralized Routing Protocol for Lifetime and Energy Optimization using GA and LSPI (CRPLEOGALSPI) for WSNs.

6.2 METHODOLOGY

This section presents the network model, the design, and the implementation of the CRPLEOGALSPI.

The design of the proposed protocol is carried out by first representing the WSN that consists of a set of sensor nodes and a sink as a weighted graph. After the network initialization, the sink constructs the possible MSTs that are used as the routing tables (RTs) using the GA-based MSTs presented in Section 5.2.1. The construction of the MSTs is repeated by the sink after the death of sensor node(s) during the round of data transmission until the network graph is disconnected. The sink then uses LSPI to learn the optimal or near-optimal MST(s) during the round of data transmission so as to maximize the network lifetime while minimizing the network energy consumption. The performance analysis of the CRPLEOGALSPI is carried out by means of a simulation using the network lifetime, the number of alive nodes (NAN), network energy consumption, and computation time as performance metrics. The novelty of this chapter is as follows:

(i) Formulation of a reward function for the joint optimization of the lifetime and energy consumption for WSNs.

(ii) Design of a centralized routing protocol using a GA and an LSPI for WSNs to improve their lifetimes and energy consumption performances.

6.2.1 *Network Model*

The network model of CRPLEOGALSPI is the same as that of CRPLOGAQL as presented in Section 4.2.1.

6.2.2 *A Centralized Routing Protocol for Lifetime and Energy Optimization using LSPI*

After the network initialization, the sink generates possible distance-based MSTs using the GA MSTs algorithm presented in Chapter 5. The generated MSTs are used as the routing tables (RTs) by the CRPLEOGALSPI. The optimization problem addressed by CRPLEOGALSPI is to maximize the network lifetime while minimizing the network energy consumption in the WSNs. The optimization problem is formulated as a bi-objective mixed integer programming defining the objective function and the set of constraints.

$$\max_{v \in V}\left\{ESRE_v\right\} + \min_{v,u \in V}\left\{\sum_{u=0}^{|V|}\sum_{v=0}^{|V|} e_{u,v}EC_{u,v}\right\} subject\ to \tag{6.1a}$$

$$\sum_{u=0,u\neq v}^{|V|} e_{u,v} = 1 \tag{6.1b}$$

$$\sum_{v=0,u\neq v}^{|V|} e_{u,v} = 1 \tag{6.1c}$$

$$d_{u,v} \leq R_{max} \tag{6.1d}$$

$$e_{u,v} \in \{0,1\} \tag{6.1e}$$

Equation (6.1a) is the objective function and Equations (6.1b)–(6.1e) are the constraints. The variable $ESRV_v$ is the estimated sensor node residual energy of the vth sensor node. $e_{u,v}$ is the connection index when the uth sensor node is connected to the vth sensor node. $E_{u,v}$ is the energy consumption when the uth sensor node communicates with the vth sensor node. $d_{u,v}$ is the link distance between the uth sensor node and the vth sensor node. R_{max} is the network's maximum transmission radius.

The first term of the objective function in Equation (6.1a) is used to maximize the network lifetime, while the second term is used to minimize the network energy consumption. The constraints in Equation (6.1b) and (6.1c) ensure that the routing path is a tree. The constraint in Equation (6.1d) enables transmission between two connected sensor nodes whose link distance is less than or equal to the maximum network transmission radius. The constraint in Equation (6.1e) provides for the connectivity between any two edges in the network graph.

Therefore, the CRPLEOGALSPI seeks to find the RT that maximizes the network lifetime while minimizing the network energy consumption using LSPI. The network lifetime considered in designing the CRPLEOGALSPI is the time for the network graph to be disconnected. That is the time taken for an alive sensor node not to find a path to reach the sink for data transmission. The CRPLEOGALSPI is deployed at the sink. The state space and action space of the agent is the MSTs generated by the GA MSTs algorithm.

For a current MST, s being used by the sink in receiving data from the sensor nodes, the agent's action is to choose another MST, a that maximizes the network lifetime while minimizing the network energy consumption. The next state of the agent, $\hat{s}$ is the same

as the choosing action in the current learning episode, that is $P(\hat{s}|s,a) = 1$. The features considered in designing the basic functions of the CRPLEOGALSPI for the state-action pair are the maximum energy consumption of the sensor nodes, $EC_{max}(s,a)$, and the sum energies consumption of the sensor nodes, $EC_{sum}(s,a)$, when using a to receive data packets by the sink.

The first basic function is used to model the maximization of the network lifetime, while the second basic function is used to model the minimization of the network energy consumption. The collection of the basic functions for the state-action pair (s,a) is therefore given as $\varphi(s,a) = \left\{ EC_{max}(s,a), EC_{sum}(s,a) \right\}$.

The Q-values of the state-action pair are approximated using Equation 6.2

$$\hat{Q}(s,a) = w_1 EC_{max}(s,a) + w_2 EC_{sum}(s,a) = w\varphi(s,a)^T \tag{6.2}$$

where w_1 and w_2 are the weights associated to the basic functions $EC_{max}(s,a)$ and $EC_{sum}(s,a)$, respectively. w is the weight matrix of size 2×1 and $\varphi(s,a)^T$ is the transpose of the basic function matrix of size 1×2.

The weight, w in the Equation 6.2 is approximated using a set of samples $\mathcal{D} = \left\{ (s_i, a_i, r_i, \hat{s_i}) | i = 1, 2..., M \right\}$ as given in Equation 6.3.

$$\hat{w} = \hat{X}^{-1}\hat{y} \tag{6.3}$$

The set of samples $\mathcal{D}$ is gotten by taking a random action in a given state. The reward earned is modeled as in Equation 6.4

$$r_i = \max_{v \in V}\{EC_v\} + \text{sum}_{v \in V}\{EC_v\} \tag{6.4}$$

where EC_v is the energy consumption by the v^{th} sensor node.

The algorithm used to generate the random samples by the sink is given in Algorithm 14.

Algorithm 14 Samples Generation Algorithm

Input: List of RTs, Maximum number of samples, $|\mathcal{D}|_{max}$, Number of iterations, N
Output: Set of samples, $\mathcal{D}$

$\mathcal{D} = \{\}$
Initialize a random RT as s_0
for $i = 1$ to N **do**
  $s_i = s_o$
  The agent Choose a random RT, a_i
  The agent receives a reward, r_i using Equation 6.4
  $s_i \leftarrow s_o$
  $\hat{s_i} = s_o$
  if $(s_i, a_i, \hat{s_i}, r_i) \notin \mathcal{D}$ **then**
    $\mathcal{D} \leftarrow (s_i, a_i, \hat{s_i}, r_i)$
  end if
  if $|\mathcal{D}| = |\mathcal{D}|_{max}$ **then**
    break
  end if
end for
Return $\mathcal{D}$

Assuming $\hat{X}^0 = 0$ and $\hat{y}^0 = 0$ initially, Therefore, Equation 6.5 and Equation 6.6 will give the approximated values of $\hat{X}^{i+1}$ and $\hat{y}^{i+1}$, respectively of a new sample $(s_i, a_i, r_i, \hat{s_i})$

$$\hat{X}^{i+1} \leftarrow \hat{X}^i + \varphi(s_i, a_i)\left[\varphi(s_i, a_i) - \gamma\varphi(\hat{s_i}, \underset{a \in A}{\operatorname{argmin}}(\hat{s_i}, a))\right]^T \tag{6.5}$$

where γ is the discount factor.

$$\hat{y}^{i+1} \leftarrow \hat{y}^i + \varphi(s_i, a_i) r_i \tag{6.6}$$

The network lifetime is maximized while minimizing the network energy consumption by selecting the RT that has a minimum Q-value using the epsilon-greedy technique. Therefore, given a number, $r \in (0, 1)$ generated randomly in each episode and a likelihood epsilon value, $\epsilon \in [0, 1]$, the learning agent chooses its action in each round using the policy given in Equation 6.7.

$$a_t = \begin{cases} \underset{a}{\operatorname{argmin}}(\varphi(s, a)^T w), \; if \; r < 1 - \epsilon \\ Random \; action, \; otherwise. \end{cases} \tag{6.7}$$

The proposed CRPLEOGALSPI for finding the optimal RT at each stage of network graph building with a view to maximizing the time taken for the network graph to be disconnected and minimizing the network energy consumption is given in Algorithm 15.

Algorithm 15 CRPLEOGALSPI

Input: $G(V, E)$, Basic Functions, φ, γ, ϵ, Learning round (L), stopping criterion, ε
Output: Optimal RT(s)

1: Sink executes **Algorithm 12** to generate List of RTs
2: Sink collect set of samples, $\mathcal{D}$ with **Algorithm 14**
3: Initialize s_0 as a random RT
4: Initialize $\hat{X}^0 \leftarrow 0$ and $\hat{y}^0 \leftarrow 0$
5: Initialize weight, $w_o \leftarrow 0$
6: $\hat{w} \leftarrow w_o$
7: **repeat**
8: $w \leftarrow \hat{w}$
9: **for** $(s_i, a_i, r_i, \hat{s_i}) \in \mathcal{D}$ **do**
10: $\hat{X}^{i+1} \leftarrow \hat{X}^i + \varphi(s_i, a_i)\left[\varphi(s_i, a_i) - \gamma\varphi(\hat{s_i}, \underset{a \in A}{\operatorname{argmin}}(\hat{s_i}, a))\right]^T$
11: $\hat{y}^{i+1} \leftarrow \hat{y}^i + \varphi(s_i, a_i) r_i$
12: **end for**
13: $\hat{w} = \hat{X}^{-1}\hat{y}$
14: **until** $||w - \hat{w}|| < \varepsilon$
15: **return** w
16: **for** $t = 1$ to L **do**
17: $s_t = s_o$
18: Sink chooses an RT using Equation 6.7
19: Sink receives data from the sensors using the chosen RT.
20: Updates s_o as the current RT.
21: **if** Any sensor node dies, **then**
22: Delete the dead sensor node(s) from $G(V, E)$
23: Delete edges connected to the dead sensor node(s)
24: Rebuild $G(V, E)$
25: **if** $G(V, E)$ is connected, **then**
26: Do steps 1 to 20
27: **else**
28: *break*
29: **end if**
30: **end if**
31: **end for**

6.3 SIMULATION AND RESULTS DISCUSSION

The performance analysis of the proposed routing protocol is achieved by simulations using the performance metrics of network lifetime, number of alive sensor nodes (NAN), energy consumption, and computation time. These metrics of the CRPLEOGALSPI are compared with that of the CRPLOGAQL as a means of validation. The network lifetime is computed as the time taken for the sink not to be reachable by alive sensor node(s). The NAN is the number of alive sensor nodes in the network lifetime.

The CRPLEOGALSPI and CRPLOGAQL are coded with Python 3.8 under the "PyCharm" development environment. The graphical structure of the WSN is implemented using the Python networkx module. The Python code is executed on the SLURM (Simple Linux Utility for Resource Management) cluster on the IRIT's OSIRIM platform. The

Computer nodes of the OSIRIM platform adopted are the 4 AMD EPYC 7402 bi-processor computing nodes at 2.8 GHz, with 48 processors and 512 GB of RAM each. These nodes enable more than 24 threads and/or 192 GB of RAM for the same process.

The deployed network graph of the WSN is the same as in Figure 3.1. The simulation parameters used to carry out the performance analysis are shown in Table 6.1

Table 6.1: Simulation Parameters for CRPLEOGALSPI

Parameters	**Values**
Number of sink	1
Number of sensors	100
Area of deployment	1000 m × 1000 m
Deployment of Sensor node	Random
Sink coordinate	$(500, 500)$
Communication range	150 m
Bandwidth	1 $kbps$
Data packet size	1024 bits
Initial sensor energy	1 J to 10 J, 10 J to 100 J
Packet generation rate	1 $/s$ to 10 $/s$
Discount factor	0.9
Epsilon	0.1
Sample size	100
Number of generations	1000
Crossover rate	0.01
Mutation rate	1

6.3.1 *Performance Comparison of CRPLEOGALSPI with CRPLOGAQL for Homogeneous Sensor Nodes IRE and PGR*

The number of alive sensor nodes of the WSN at each round of data transmission of the proposed routing protocol, CRPLEOGALSPI, as compared with CRPLOGAQL when the initial sensor nodes energies and the packet generation rate of the sensor nodes are set arbitrarily at 10 J and 1 $/s$, respectively, as shown in Figure 6.1.

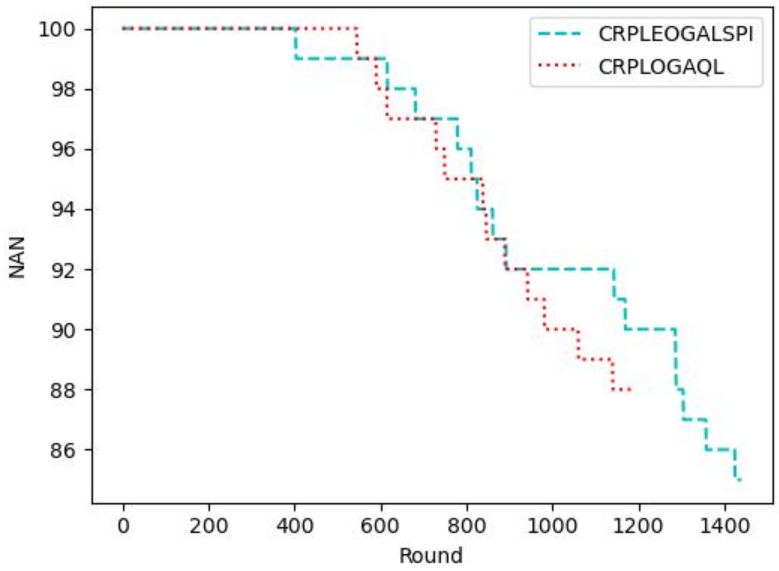

Figure 6.1: Number of alive Sensors with Round of Data Transmission.

As seen in Figure 6.1, the number of alive sensor nodes of both protocols remains constant for certain rounds of data transmission and decreases subsequently. This behavior continues for both protocols until the network graph is disconnected. The deaths of the sensor nodes are because of the depletion of the energy of the sensor nodes as the round of data transmission increases. The time for the sink not to be reachable by alive sensor nodes of CRPLEOGALSPI is compared with CRPLOGAQL for increasing sensor nodes' initial energy as shown in Figure 6.2.

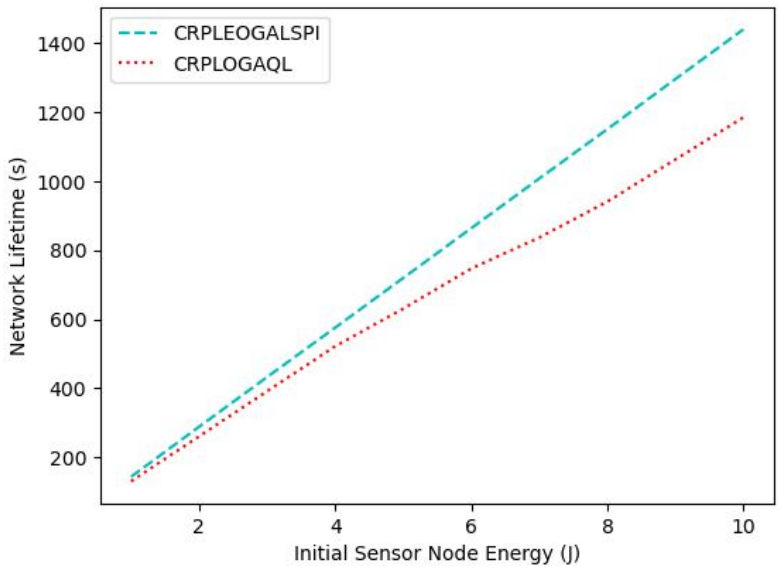

Figure 6.2: Network lifetime with Initial Node Energy.

Consequently, on average, it took CRPLEOGALSPI more rounds of data transmission for the sink not to be reachable by alive sensor nodes. This is because CRPLEOGALSPI converges faster to the RT, maximizing the network lifetime while minimizing the network energy consumption compared to CRPLOGAQL. This shows that the LSPI used in implementing CRPLEOGALSPI utilizes data effectively and efficiently compared to the

baseline Q-learning used to implement the CRPLOGAQL. The network lifetime increases with the increase in the initial sensor node energy. This is because the lifetime of a sensor node is proportional to the residual energy of the sensor node.

The proposed CRPLEOGALSPI has an improved network lifetime performance of 18.04% when compared to CRPLOGAQL with increasing sensor node residual energy. This is because CRPLEOGALSPI utilizes LSPI which is more data efficient and convergences faster to the RT which balances the energy consumption among the sensor nodes and minimizes the network energy consumption as shown in Figure 6.3.

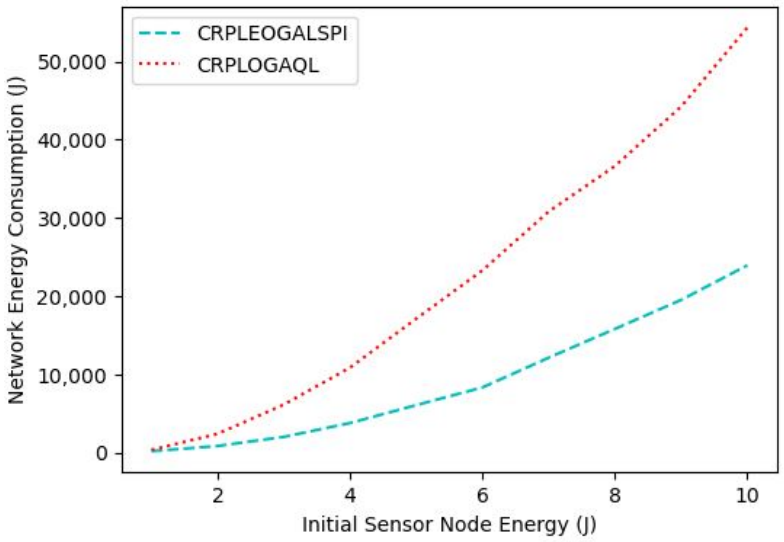

Figure 6.3: Network Energy Consumption with Initial Node Energy.

This is against the CRPLOGAQL that utilizes the baseline Q-learning that takes larger episodes to converge to the RT that optimizes the network lifetime due to the many RTs. The network energy consumption for both protocols increases with the increasing initial energy of the sensor nodes. This is because the more the initial sensor nodes' energies, the more rounds of data transmission it takes for the sensor nodes to deplete their energies. Subsequently, the CRPLEOGALSPI has a reduced energy consumption of 58.96% when compared to CRPLOGAQL for increasing the initial energy of the sensor nodes. This is because CRPLEOGALSPI optimizes jointly the network lifetime and the network energy consumption using LSPI. This is against the CRPLOGAQL which optimizes only the network lifetime using Q-learning. This also implies that CRPLEOGALSPI has a reduced CO_2 footprint when compared to CRPLOGAQL for increased initial sensor node energy. This is because CO_2 footprint is directly proportional to the network energy consumption. However, the improved performance in the network lifetime and network energy consumption of the proposed protocol comes with an increased computation time of about ten times when compared to CRPLOGAQL for an increased initial sensor node energy, as shown in Figure 6.4.

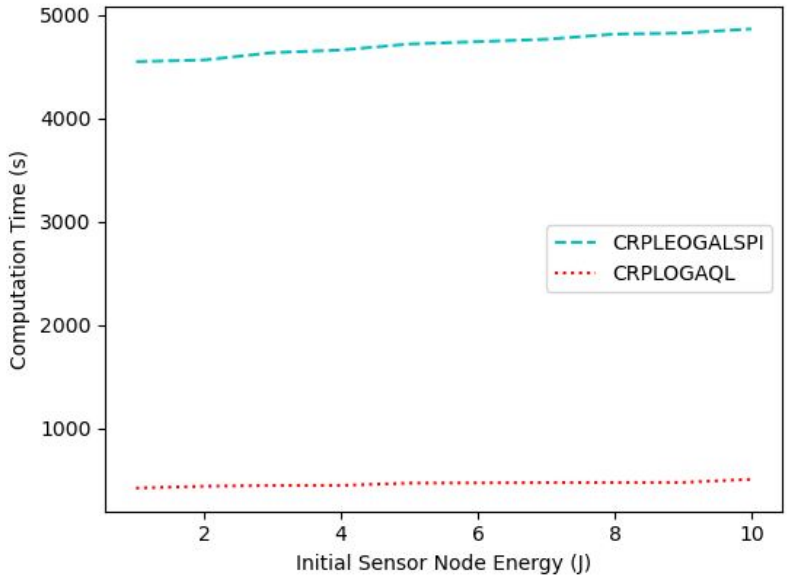

Figure 6.4: Computation Time with Initial Node Energy.

This is because CRPLEOGALSPI requires the collection of samples to learn the optimal path each time the network is rebuilt because of the death of sensor node(s).

The time for the sink not to be reachable by alive sensor nodes of CRPLEOGALSPI is compared with CRPLOGAQL for increasing sensor nodes' packet generation rate as shown in Figure 6.5. The network lifetime decreases with the increased packet generation rate of the sensor nodes. This is because the network lifetime of a sensor node is inversely proportional to the number of packets transmitted by the sensor node. The proposed CRPLEOGALSPI has an improved network lifetime performance of 14.91% when compared to CRPLOGAQL with an increasing packet generation rate of the sensor nodes.

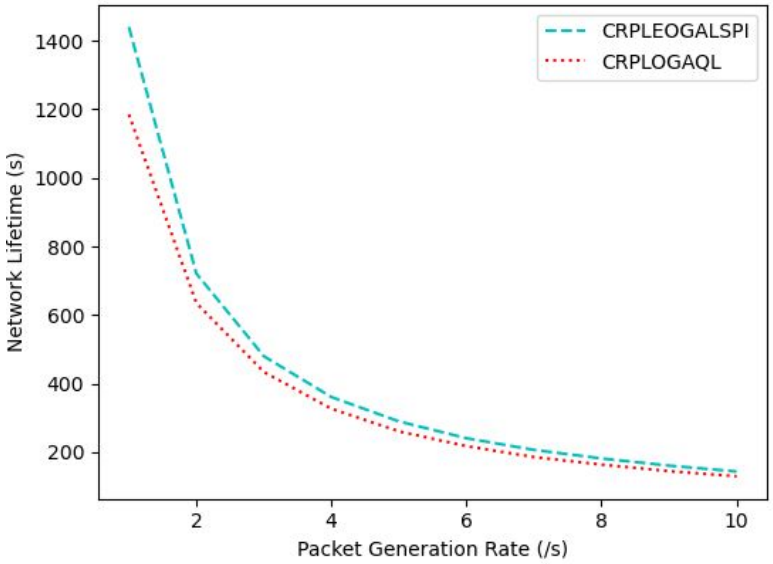

Figure 6.5: Network Lifetime with Packet Generation Rate.

This is because CRPLEOGALSPI utilizes LSPI which is more data efficient and convergences faster to the RT which balances the energy consumption among the sensor nodes and minimizes the network energy consumption as shown in Figure 6.6.

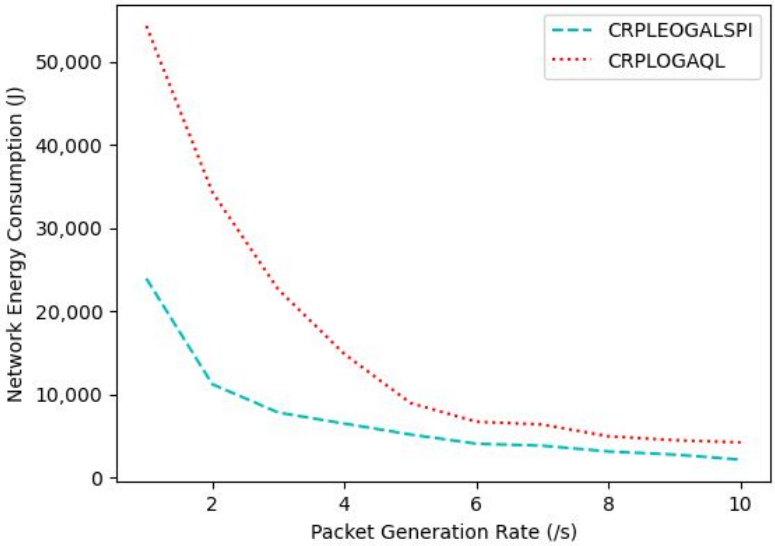

Figure 6.6: Network Energy Consumption with Packet Generation Rate.

This is against the CRPLOGAQL that utilizes the baseline Q-learning that takes larger episodes to converge to the RT that optimizes the network lifetime due to the many RTs. The network energy consumption for both protocols decreases with the increasing packet generation rate of the sensor nodes. This is because the more the number of data packets generated and transmitted by the sensor nodes, the fewer rounds of data transmission it takes for the sensor nodes to deplete their energies. Subsequently, the CRPLEOGALSPI has a reduced energy consumption of 56.36% when compared to CRPLOGAQL for increasing the packet generation rate of the sensor nodes. This is because CRPLEOGALSPI optimizes jointly the network lifetime and the network energy consumption using LSPI. This is against the CRPLOGAQL which optimizes only the network lifetime using Q-learning. This is contrasted with the CRPLOGAQL, which optimizes only the network lifetime using Q-learning. This also implies that the CRPLEOGALSPI has a reduced CO_2 footprint when compared to the CRPLOGAQL for an increased sensor node packet generation rate. This is because that CO_2 footprint is directly proportional to the network energy consumption.

But, the improved performance in the network lifetime and network energy consumption of the proposed protocol comes with an increased computation time of about ten times that of the CRPLOGAQL for an increased packet generation rate of the sensor node, as shown in Figure 6.7. This is because the CRPLEOGALSPI requires the collection of samples and iteration over the collected samples to learn the optimal policy each time the network is rebuilt because of the death of sensor node(s).

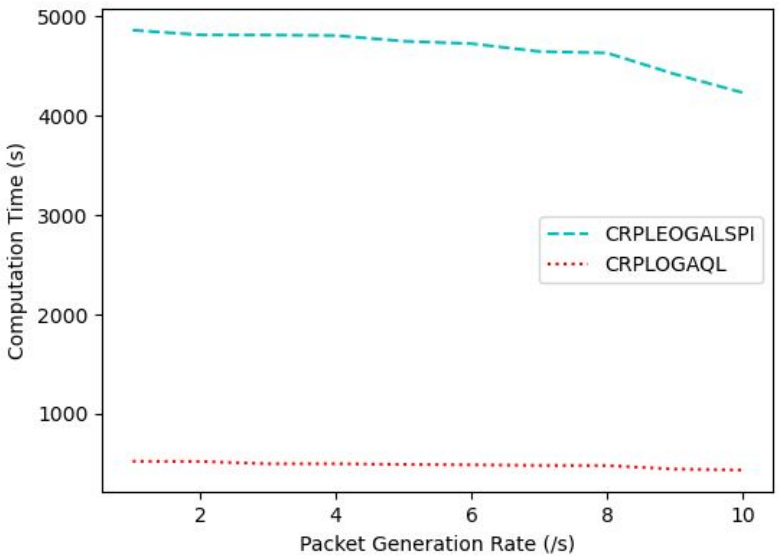

Figure 6.7: Computation Time with Packet Generation Rate.

6.3.2 *Performance Comparison of CRPLEOGALSPI with CRPLOGAQL for Heterogeneous PGR and Homogeonous IRE of Sensor Nodes*

The performance of CRPLEOGALSPI is compared with that of CRPLOGAQL for heterogeneous packet generation rate and homogenous initial residual energy of sensor nodes. This is achieved by each sensor node choosing a random PGR among $1\ /s$ and $10\ /s$ inclusive of the multiples of $1\ /s$ to be used in every round of the data transmission until the network is disconnected. To accommodate the large number of packets in the network and the number of rounds required by both protocols to learn the best paths to maximize the network lifetime, the initial residual energy of the sensor nodes are the same and increased arbitrarily from $10\ J$ to $100\ J$. This is because the centralized agent mostly for CRPLOGAQL requires a long term of learning to find the optimal paths due to the large state space and action space. Figure 6.8 shows the comparison of the number of alive nodes in each round of data transmission of CRPLEOGALSPI with that of the CRPLOGAQL when the initial residual energy of the sensor nodes is set constant arbitrarily at $100\ J$ and the packet generation rate of the sensor nodes are heterogenous as stated. The number of alive sensor nodes of both routing protocols decreases as the round of data transmission increases for the scenario of heterogeneous traffic. The decrease in the number of alive sensor nodes is due to the depletion of the energy sources of the sensor nodes as the round of data transmission increases.

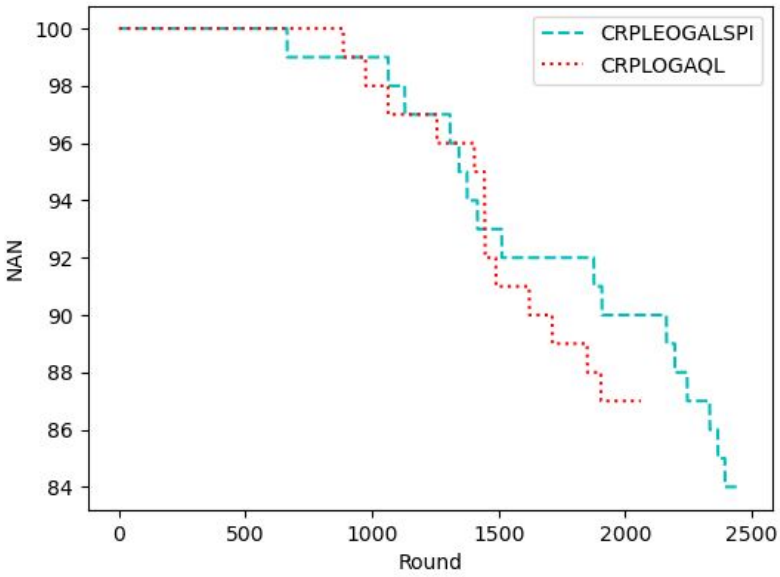

Figure 6.8: NAN with Round for Heterogeneous Sensor Nodes PGR

As seen in Figure 6.8, mostly the number of rounds of data transmission for CRPLEOGALSPI between the depletion of the energy of the sensor nodes on average is higher than that of CRPLOGAQL for the heterogeneous traffic at the sensor nodes. Therefore the CRPLEOGALSPI has a higher network lifetime when compared with CRPLOGAQL for heterogeneous traffic. This is because CRPLEOGALSPI converges faster to the optimal MST(s), maximizing the network lifetime while minimizing the network energy consumption compared to CRPLOGAQL. Also, as shown in Figure 6.8 CRPLEOGALSPI has 84 alive sensor nodes when the network graph is disconnected, while the CRPLOGAQL has 87 alive sensor nodes before the sink is no longer reachable for data transmission. This implies that CRPLEOGALSPI learns the optimal routes to transmit the different number of packets generated by each sensor node to the sink in a way that the sensor nodes energy consumption is balanced as the round of data transmission increases, minimizing the energy consumption by each sensor node, and prolonging the network lifetime. Figure 6.9 shows the network lifetime of CRPLEOGALSPI and CRPLOGAQL for heterogeneous traffic at the sensor nodes when the initial residual energy is homogenous and increased from $10J$ to $100J$.

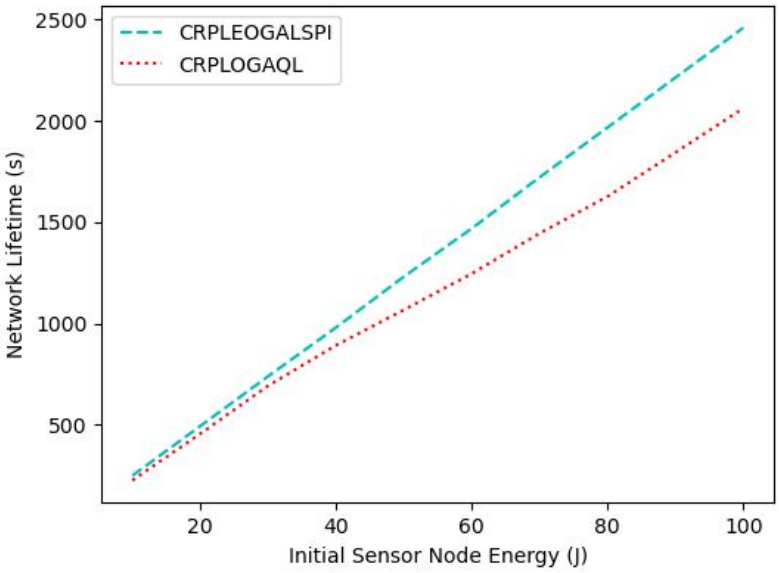

Figure 6.9: Network Lifetime with increasing Initial Sensor Node Energy for Heterogeneous Sensor Nodes PGR

As seen in Figure 6.9 as the initial residual energy of the sensor nodes increases with the heterogeneous traffic, the network lifetime of both protocols also increases. The higher the initial residual energy of the sensor nodes, the more round of data transmission it takes each sensor node to deplete its energy source. Subsequently, as the initial residual energy of the sensor nodes increases, CRPLEOGALSPI performs better than the CRPLOGAQL in terms of network lifetime and network energy consumption. This is attributed to CRPLEOGALSPI learning the routing paths that jointly maximize the network lifetime while minimizing the network energy consumption using LSPI which is more efficient than Q-Learning. The higher the initial residual energy of the sensor nodes, the more the number of rounds the sink in CRPLEOGALSPI utilizes the optimal MST(s) to receive the different packets from the sensor nodes. This causes CRPLEOGALSPI to consume less energy when compared with CRPLOGAQL as the initial residual energy of the sensor nodes increases as shown in Figure 6.10.

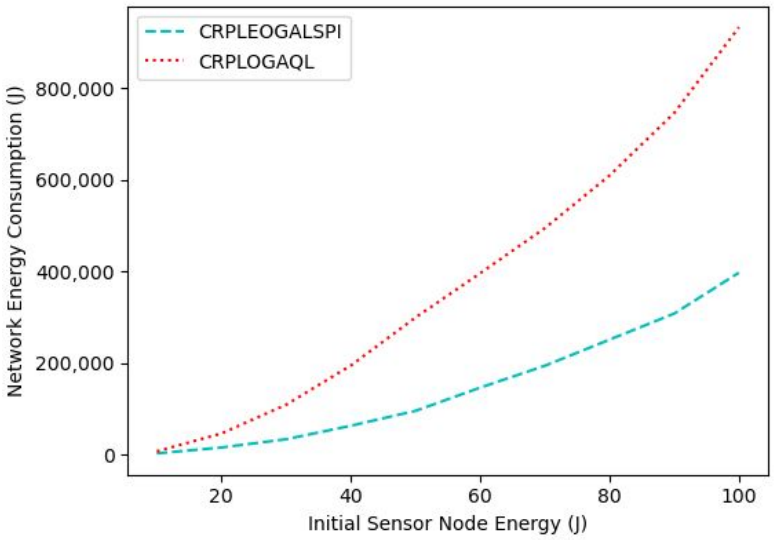

Figure 6.10: Network Energy Consumption with increasing Initial Sensor Node Energy for Heterogeneous Sensor Nodes PGR

The CRPLEOGALSPI has an increased network lifetime performance of 17.03% and a reduced network energy consumption of 60.55% when compared with CRPLOGAQL for increasing homogenous initial residual energy and different packet generation rate of the sensor nodes.

However, the improved performance in the network lifetime and network energy consumption of the CRPLEOGALSPI comes with an increased computation time of 84.30% when compared to CRPLOGAQL as shown in Figure 6.11.

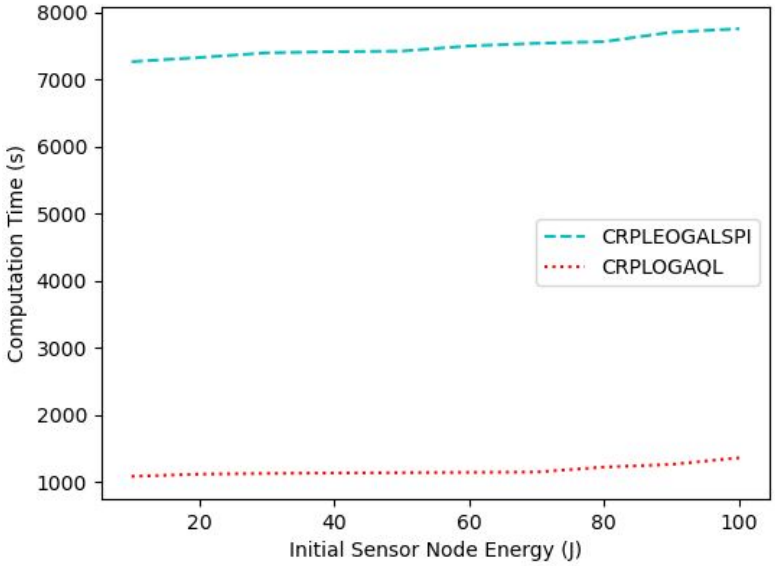

Figure 6.11: Computation Time with increasing Initial Sensor Node Energy for Heterogeneous Sensor Nodes PGR

6.3.3 *Performance Comparison of CRPLEOGALSPI with CRPLOGAQL for Heterogeneous IRE and Homogeonous PGR of Sensor Nodes*

The NAN, network lifetime, network energy consumption, and computation time performance of CRPLEOGALSPI is compared with that of CRPLOGAQL for heterogeneous initial residual energy and homogenous packet generation rate of sensor nodes. Each sensor node has different initial residual energy that is randomly chosen between $10J$ and $100J$ inclusive by the multiple of $10J$ while the packet generation rate of the sensor nodes is fixed and arbitrarily increased from $1/s$ to $10/s$. Figure 6.12 shows the comparison of the NAN in each round of data transmission of CRPLEOGALSPI with that of the CRPLOGAQL when the packet generation rate of the sensor nodes is set constant arbitrarily at $1/s$ and the initial residual energy of the sensor nodes are heterogeneous.

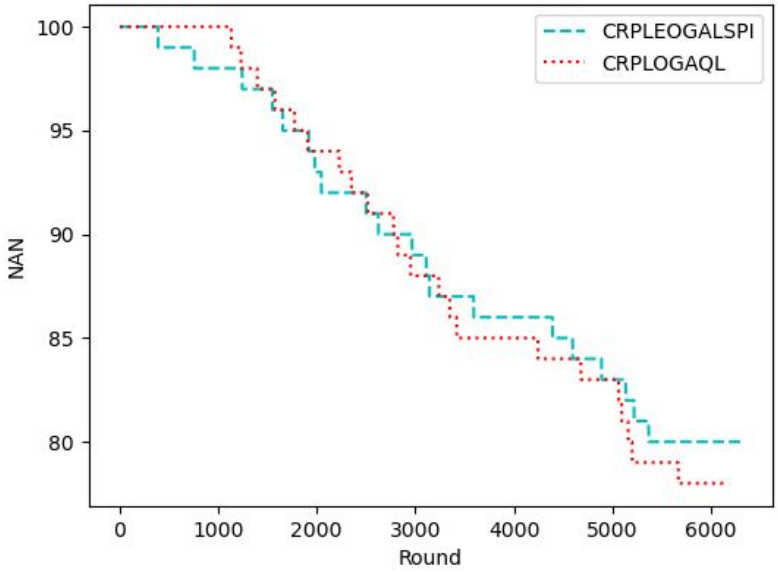

Figure 6.12: NAN with Data Transmission Round for Heterogeneous Sensor Nodes IRE

The network lifetime and NAN performance of CRPLEOGALSPI are slightly higher than that of LACQRP when the initial residual energy of the sensor nodes is heterogeneous. This is attributed to CRPLEOGALSPI learning the routing paths that jointly maximize the network lifetime while minimizing the network energy consumption using LSPI which is more efficient than Q-Learning utilize in CRPLOGAQL. This makes the round of data transmission on average between the reconstruction of the network graph by the sink while using CRPLEOGALSP to be relatively high when compared with CRPLOGAQL. The CRPLEOGALSPI has 80 alive sensor nodes when the network graph is disconnected, and the CRPLOGAQL with 78 alive sensor nodes before the sink is no longer reachable for data transmission. This resulted in CRPLEOGALSPI having a 2.56% improvement in NAN performance compared to CRPLOGAQL. Therefore the CRPLEOGALSPI makes the network graph to be disconnected less when compared with CRPLOGAQL. Figure 6.13 shows the network lifetime of CRPLEOGALSPI and CRPLOGAQL for heterogeneous initial residual energy at the sensor nodes when the packet generation rate is homogenous and increases from $1/s$ to $10/s$.

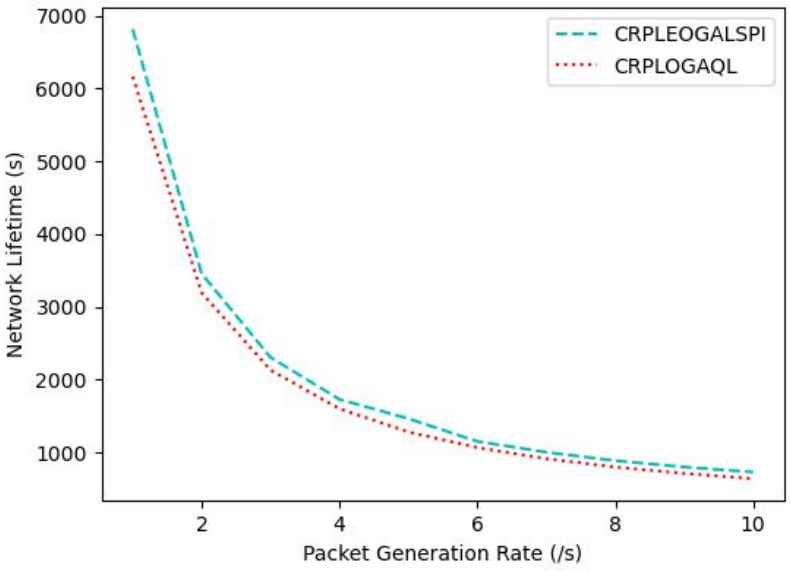

Figure 6.13: Network Lifetime with increasing Sensor Node PGR for Heterogeneous Sensor Nodes Initial Energy

The CRPLEOGALSPI has a high network lifetime performance of 9.94% and low network energy consumption of 72.03% when compared with CRPLOGAQL for increasing homogenous packet generation rate and heterogenous initial residual energy of the sensor nodes as shown in Figure 6.13 and Figure 6.14, respectively.

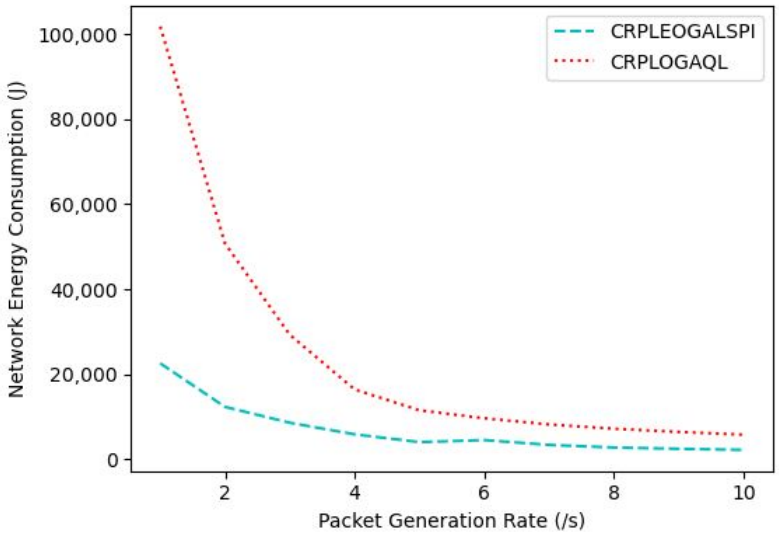

Figure 6.14: Network Energy Consumption with increasing Sensor Node PGR for Heterogeneous Sensor Nodes Initial Energy

However, the improved performance in the network lifetime and network energy consumption of the CRPLEOGALSPI comes with an increased computation time of 79.26% when compared to CRPLOGAQL as shown in Figure 6.15.

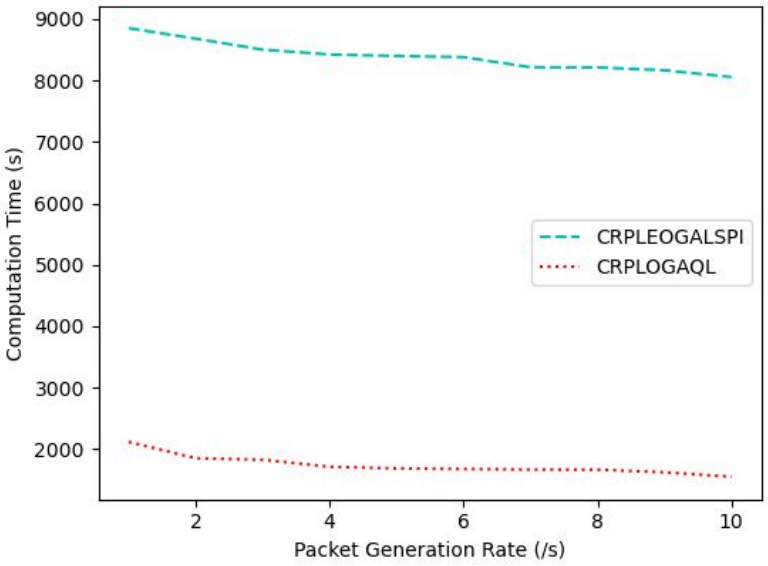

Figure 6.15: Computation Time with increasing Sensor Node PGR for Heterogeneous Sensor Nodes Initial Energy

6.3.4 *Performance Comparison of CRPLEOGALSPI with CRPLOGAQL for Heterogeneous PGR and IRE of Sensor Nodes*

The NAN, network lifetime, network energy consumption, and computation time performance of CRPLEOGALSPI is compared with that of CRPLOGAQL when the sensor nodes IRE and PGR are heterogeneous. This is achieved by choosing the IRE of the sensor nodes randomly among 10J and 100J inclusive of the multiples of 10J. Also, the PGR of each sensor node is randomly chosen among 1/s and 10/s inclusive of the multiples of 1/s. Figure 6.16 and Figure 6.17 show the NAN and network lifetime performance of CRPLEOGAQL when compared with CRPLOGAQL, respectively.

As shown in Figure 6.16, the network disconnected relatively slower with CRPLEOGALSPI when compared with CRPLOGAQL. Also, the CRPLEOGALSPI has 80 alive sensor nodes when the network graph is disconnected, while the CRPLOGAQL has 78 alive sensor nodes before the sink is no longer reachable for data transmission. This resulted in CRPLEOGALSPI having 2.56% and 4.31% improvement in NAN and network lifetime, respectively performance compared to CRPLOGAQL. This is attributed to CRPLEOGALSPI learning the routing paths that jointly maximize the network lifetime while minimizing the network energy consumption using LSPI which is more efficient than Q-Learning for the scenario of heterogeneous traffic and IRE of sensor nodes.

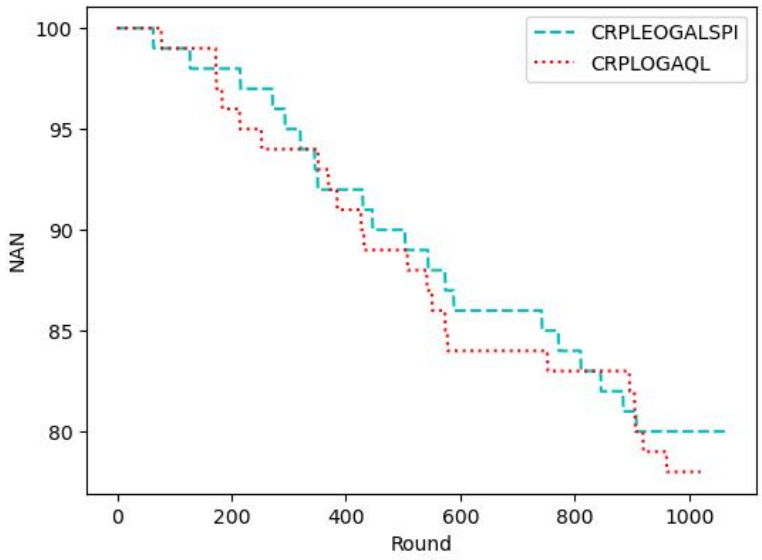

Figure 6.16: NAN with Data Transmission Round for Heterogeneous Sensor Nodes IRE and PGR

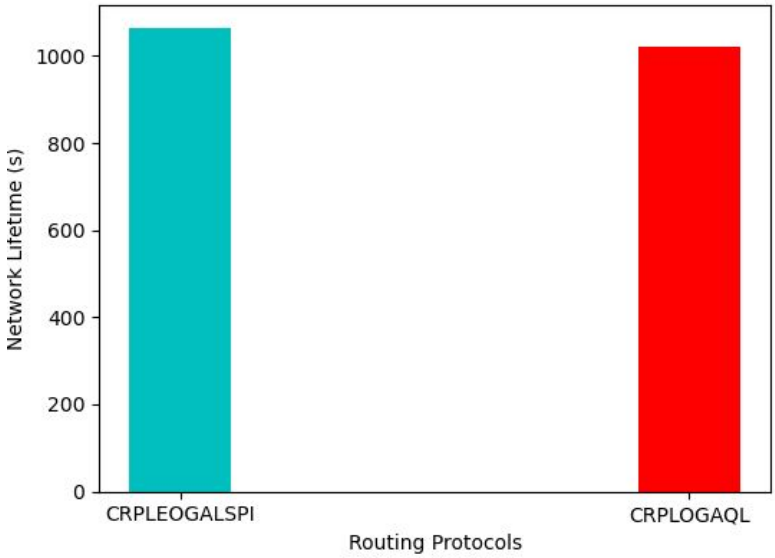

Figure 6.17: Network Lifetime for Heterogeneous Sensor Nodes IRE and PGR

This causes CRPLEOGALSPI to consume 37.19% less network energy with respect to CRPLOGAQL as shown in Figure 6.18.

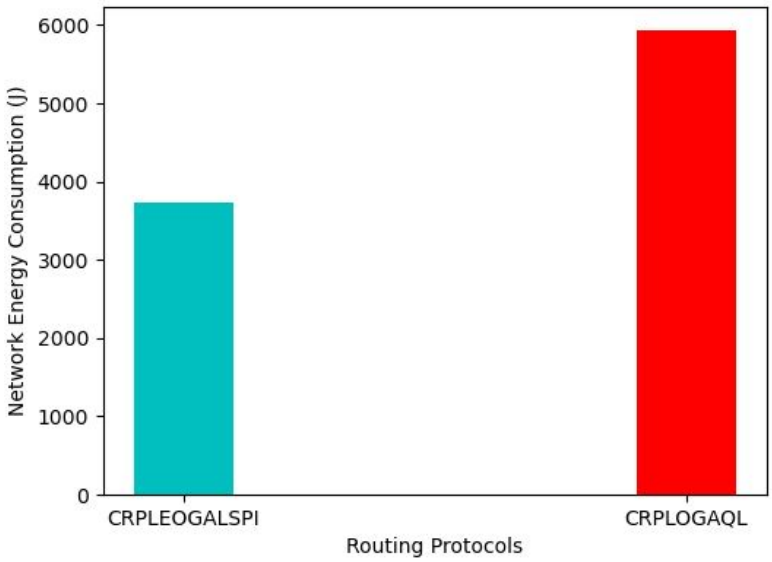

Figure 6.18: Network Energy Consumption for Heterogeneous Sensor Nodes IRE and PGR

However, the improvement of the NAN, network lifetime, and network energy consumption performance of CRPLEOGALSPI to CRPLOGAQL comes with a degradation in the computation time of 86.45% as shown in Figure 6.19.

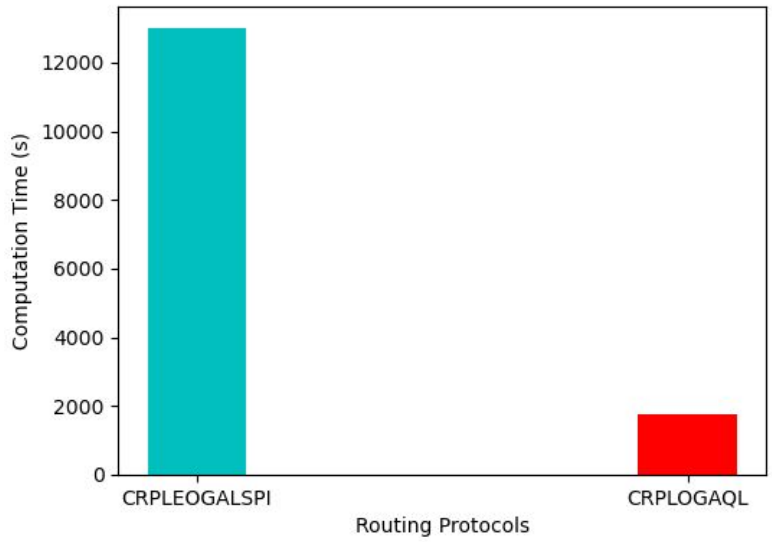

Figure 6.19: Computation Time for Heterogeneous Sensor Nodes IRE and PGR

6.4 CONCLUSION

This chapter presented the design of a centralized routing protocol for a lifetime and energy optimization using a GA and LSPI for WSNs. The sink generates the routing tables of the network graph in polynomial time using a GA. The LSPI deployed at the sink learns the optimal routing path for a lifetime and energy optimization at each stage of the network graph, building after the death of sensor nodes. This leads to the maximization of the time taken for the sink not to be reachable by the alive sensor nodes while minimizing the network energy consumption. The centralized routing protocol for

a lifetime and energy optimization using a GA and LSPI improve network lifetime and energy consumption when compared with the Centralized Routing Protocol for Lifetime Optimization using Genetic Algorithm and Q-learning. This is because the centralized routing protocol for lifetime and energy optimization using GA and LSPI converges faster to the optimal routing path and is not sensitive to parameter settings. However, the improved performance in the network lifetime and network energy consumption of the centralized routing protocol for a lifetime and energy optimization using a GA and LSPI comes with an increased computation time when compared with the Centralized Routing Protocol for Lifetime Optimization using Genetic Algorithm and Q-learning.

7 CONCLUSION AND FUTURE RESEARCH

7.1 SUMMARY OF CONTRIBUTIONS

Wireless Sensor Networks (WSNs) consist of small-size, low-cost, low-processing devices with a limited battery capacity that seeks to transmit information to a sink or base station (BS) through multi-hop routing. This has made the sensor nodes' energy-efficient utilization a major challenge in the design of WSNs. This is because the network lifetime is determined by the sensor nodes' limited energy sources whose replacement or recharging is almost impossible due to the mostly deployment of the sensor nodes in harsh environments. Intelligent routing protocols are required because of the need to achieve optimal information delivery in dynamic, challenging, and complex WSNs environments. This has resulted in the use of RL algorithms to design energy-efficient routing protocols in WSNs so that the network lifetime is improved.

Chapter 3 entailed the performance analysis of the impact of learning rate and discount factor on distributed RL-based routing protocols network lifetime and energy consumption for WSNs. The network lifetime and energy consumption are better for a low learning rate and high discount factor. This resulted in an improved lifetime and average energy consumption per round of the distributed RL-based lifetime-aware routing protocols. The lower the learning rate, the better the preservation of the learned optimal routing path. The higher the discount factor, the more the learning process utilizes the cumulative reward. Thus the learning agent will choose the best routing paths which will result in an increased network lifetime because of reduced energy consumed by the sensor nodes in sending data packets to their next forwarder. However, the convergence rate of the distributed RL-based lifetime-aware routing protocols is low because each sensor node learns the next forwarder locally and needs many iterations to select the optimal next forwarder, this leads to the degradation of the network lifetime and average energy consumption per round.

Chapter 4 presented the design of a lifetime-aware centralized Q-routing protocol to improve the convergence rate, network lifetime, and average energy consumption when compared with the RL-distributed routing protocols. The sink of the designed centralized routing protocol also acted as the controller that has global knowledge of the network information and enables the generation of all possible distance-based MSTs which are used as routing tables. Q-learning is deployed at the sink to learn the routing path that maximizes the lifetime for the first sensor node to deplete its energy source. The proposed protocol learns the best MST(s) that optimize the network lifetime and reduces the average network energy consumption of the WSNs. The designed LACQRP has a better convergence rate, network lifetime, and average energy consumption when compared with the distributed RL routing protocols of RLBR and R2LTO. However, the limitation of the LACQRP is that it depends on an algorithm that generates all MSTs of a graph. The problem of generating all MSTs of a graph is NP-hard (computational complexity is exponential).

Chapter 5 presented the design of a centralized routing protocol for lifetime optimization using GA-based MSTs and Q-Learning for WSNs to remove the NP-hardness associated with the LACQRP. The sink which contains the learning agent generated

subsets of all MSTs of the network graph in polynomial time for routing purposes using the designed GA-based MSTs. The Q-Learning deployed at the sink learned the optimal MST(s) for lifetime optimization at each stage of the network building until the sink is not reachable by alive sensor nodes. This led to the maximization of the time taken for the sink not to be reachable by alive sensor nodes. The centralized routing protocol for lifetime optimization using GA-based MSTs and Q-Learning has improved computation time when compared with the lifetime-aware centralized Q-routing protocol that uses all MSTs of the network graph. However, the centralized routing protocol for lifetime optimization using GA-based MSTs and Q-Learning suffered a reduction in the performance of the network lifetime, network energy consumption, and the number of alive nodes when the network graph is disconnected. This is because the subset of the MSTs generated by the GA-based MSTs does not contain all the optimal MST(s) and Q-Learning requires a large learning round to find the few optimal MSTs.

Finally, Chapter 6 presented the design of a centralized routing protocol for a lifetime and energy optimization using GA-based MSTs and LSPI for WSNs. The LSPI deployed at the sink learns the optimal routing path for a lifetime and energy optimization at each stage of the network graph, building after the death of sensor nodes. This leads to the maximization of the time taken for the sink not to be reachable by the alive sensor nodes while minimizing the network energy consumption. The CRPLEOGALSPI when compared with the CRPLOGAQL has improved performance in network lifetime and network energy consumption. This is because the CRPLEOGALSPI converges faster to the optimal routing paths and is not sensitive to learning rate parameter settings. However, the improved performance in the network lifetime and network energy consumption of the CRPLEOGALSPI comes with an increased computation time when compared with CRPLOGAQL.

7.2 FUTURE RESEARCH

Lifetime and energy optimization of WSNs is a key and challenging problem that requires ever-increasing attention because of the proliferation of WSN application-specific requirements. A number of possible directions for extending and improving this research work in the future are highlighted in the sequel.

(i) The RL-based optimization algorithm parameters which are learning rate, discount factor, and epsilon impact its performance significantly. In this research work, the tuning parameters are varied manually to examine the impacts on the proposed techniques. Future should consider using RL-based hyper-parameter search to have the optimal tuning parameter combinations for improving network performance.

(ii) The number of minimum spanning trees of a network graph grows exponentially with the number of nodes. For WSNs with thousands or millions of nodes, the Q-table utilized in this thesis becomes ridiculously large and impractical. Future work should consider using Deep Q-learning to save memory for the Q-table while reducing the amount of time to explore each state for very large WSNs.

(iii) Future work should consider developing a hybrid of energy-lifetime-aware RL-based centralized routing protocol and RL-based distributed routing protocols and compare the performance with the individual RL-based centralized routing protocol and RL-based distributed routing protocol.

(iv) The proposed techniques only considered RL-based lifetime and energy optimization. Future work should incorporate joint optimization of other WSNs' quality of service parameters such as latency and Packet Delivery Ratio (PDR).

(v) The performance analysis of the proposed techniques is performed using simulations. Future work can consider emulating the proposed techniques using Mininet while considering the real-world parameters of a typical WSN, and taking into consideration the cost of control packets.

PUBLICATIONS

[1] Elvis Obi, Zoubir Mammeri, and Okechukwu E Ochia. "A Lifetime-Aware Centralized Routing Protocol for Wireless Sensor Networks using Reinforcement Learning." In: *2021 17th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*. IEEE. 2021, pp. 363–368.

[2] Elvis Obi, Zoubir Mammeri, and Okechukwu E Ochia. "Centralized Routing for Lifetime Optimization Using Genetic Algorithm and Reinforcement Learning for WSNs." In: *Proceedings of the 16th International Conference on Sensor Technologies and Applications, Lisbon, Portugal*. 2022, pp. 16–20.

[3] Elvis Obi, Zoubir Mammeri, and Okechukwu E Ochia. "A Centralized Routing for Lifetime and Energy Optimization in WSNs Using Genetic Algorithm and Least-Square Policy Iteration." In: *Computers* 12.2 (2023), pp. 1–22.

BIBLIOGRAPHY

[1] Ali Forghani Elah Abadi, Seyedeh Elham Asghari, Sepideh Sharifani, Seyyed Amir Asghari, and Mohammadreza Binesh Marvasti. "A Survey on Utilizing Reinforcement Learning in Wireless Sensor Networks Routing Protocols." In: *2022 13th International Conference on Information and Knowledge Technology (IKT)*. IEEE. 2022, pp. 1–7.

[2] Ali Forghani Elah Abadi, Seyyed Amir Asghari, Mohammadreza Binesh Marvasti, Golnoush Abaei, Morteza Nabavi, and Yvon Savaria. "RLBEEP: Reinforcement-Learning-Based Energy Efficient Control and Routing Protocol for Wireless Sensor Networks." In: *IEEE Access* 10 (2022), pp. 44123–44135.

[3] Ali Abdulwahid and Muwaffaq Salih. "Wireless Sensor Networks Applications, Challenges, and Security Requirements." In: *Proceedings of 2nd International Multi-Disciplinary Conference Theme: Integrated Sciences and Technologies, IMDC-IST 2021, 7-9 September 2021, Sakarya, Turkey*, pp. 1–11.

[4] Gulnaz Ahmed, Jianhua Zou, Mian Muhammad Sadiq Fareed, and Muhammad Zeeshan. "Sleep-awake energy efficient distributed clustering algorithm for wireless sensor networks." In: *Computers & Electrical Engineering* 56 (2016), pp. 385–398.

[5] Shathee Akter and Seokhoon Yoon. "Reinforcement Learning-based Duty Cycle Interval Control in Wireless Sensor Networks." In: *International journal of advanced smart convergence* 7.4 (2018), pp. 19–26.

[6] Ian F Akyildiz, Weilian Su, Yogesh Sankarasubramaniam, and Erdal Cayirci. "Wireless sensor networks: a survey." In: *Computer networks* 38.4 (2002), pp. 393–422.

[7] Ian F Akyildiz and Mehmet Can Vuran. *Wireless sensor networks*. John Wiley & Sons, 2010.

[8] Zaher Al Aghbari, Ahmed M Khedr, Walid Osamy, Ifra Arif, and Dharma P Agrawal. "Routing in wireless sensor networks using optimization techniques: A survey." In: *Wireless Personal Communications* 111.4 (2020), pp. 2407–2434.

[9] K Al-Ani, A Abdalkafor, and A Nassar. "An overview of wireless sensor network and its applications." In: *Indonesian Journal of Electrical Engineering and Computer Science* 17.3 (2020), pp. 1480–1486.

[10] Musatafa Abbas Albadr, Sabrina Tiun, Masri Ayob, and Fahad Al-Dhief. "Genetic algorithm based on natural selection theory for optimization problems." In: *Symmetry* 12.11 (2020), pp. 1–31.

[11] Ali Alemdar and Mohamed Ibnkahla. "Wireless sensor networks: Applications and challenges." In: *2007 9th International Symposium on Signal Processing and Its Applications*. IEEE. 2007, pp. 1–6.

[12] Ahmad Ali, Yu Ming, Sagnik Chakraborty, and Saima Iram. "A comprehensive survey on real-time applications of WSN." In: *Future internet* 9.4 (2017), pp. 1–22.

[13] Muteeah Aljawarneh, Rim Hamdaoui, Ahmed Zouinkhi, Boumedyen Boussaid, and Mohamed Naceur Abdelkrim. "Energy efficiency approaches in Wireless Sensor Networks." In: *2022 IEEE 21st International Conference on Sciences and Techniques of Automatic Control and Computer Engineering (STA)*. IEEE. 2022, pp. 736–741.

[14] T Agostinho de Almeida, Akebo Yamakami, and Márcia Tomie Takahashi. "An evolutionary approach to solve minimum spanning tree problem with fuzzy parameters." In: *International Conference on Computational Intelligence for Modelling, Control and Automation and International Conference on Intelligent Agents, Web Technologies and Internet Commerce (CIMCA-IAWTIC'06)*. Vol. 2. IEEE. 2005, pp. 203–208.

[15] Tiago A Almeida, VN Souza, FMS Prado, Akebo Yamakami, and Márcia T Takahashi. "A genetic algorithm to solve minimum spanning tree problem with fuzzy parameters using possibility measure." In: *NAFIPS 2005-2005 Annual Meeting of the North American Fuzzy Information Processing Society*. IEEE. 2005, pp. 627–632.

[16] Sumayah Almuntasheri and Mohammed JF Alenazi. "Software-Defined Network-Based Energy-Aware Routing Method for Wireless Sensor Networks in Industry 4.0." In: *Applied Sciences* 12.19 (2022), pp. 1–22.

[17] Th Arampatzis, John Lygeros, and Stamatis Manesis. "A survey of applications of wireless sensors and wireless sensor networks." In: *Proceedings of the 2005 IEEE International Symposium on, Mediterrean Conference on Control and Automation Intelligent Control, 2005.* IEEE. 2005, pp. 719–724.

[18] Rocio Arroyo-Valles, Rocio Alaiz-Rodriguez, Alicia Guerrero-Curieses, and Jesús Cid-Sueiro. "Q-probabilistic routing in wireless sensor networks." In: *2007 3rd International Conference on Intelligent Sensors, Sensor Networks and Information*. IEEE. 2007, pp. 1–6.

[19] Mohammed Sulaiman BenSaleh, Raoudha Saida, Yessine Hadj Kacem, and Mohamed Abid. "Wireless sensor network design methodologies: A survey." In: *Journal of Sensors* 2020 (2020), pp. 1–13.

[20] Fazilet Lemya Benmansour and Nabila Labraoui. "A comprehensive review on swarm intelligence-based routing protocols in wireless multimedia sensor networks." In: *International Journal of Wireless Information Networks* 28.2 (2021), pp. 175–198.

[21] Pieter Beyens, Maarten Peeters, Kris Steenhaut, and Ann Nowe. "Routing with compression in wireless sensor networks: a q-learning approach." In: *Fifth European Workshop on Adaptive Agents and Multi-Agent Systems (AAMAS'05), Paris, France, March*. 2005.

[22] Tatiana Bokareva, Wen Hu, Salil Kanhere, Branko Ristic, Neil Gordon, Travis Bessell, Mark Rutten, and Sanjay Jha. "Wireless sensor networks for battlefield surveillance." In: *Proceedings of the land warfare conference*. 2006, pp. 1–8.

[23] Mahamadi Boulou, Tiguiane Yélémou, Achille Go, and Hamadoun Tall. "Energy management techniques in Software-Define Wireless Sensor Network." In: *Proceedings of the 4th edition of the Computer Science Research Days, JRI 2021, 11-13 November 2021, Bobo-Dioulasso, Burkina Faso*. 2022, pp. 1–18.

[24] Salah Eddine Bouzid, Youssef Serrestou, Kosai Raoof, and Mohamed Nazih Omri. "Efficient routing protocol for wireless sensor network based on reinforcement learning." In: *2020 5th International Conference on Advanced Technologies for Signal and Image Processing (ATSIP)*. IEEE. 2020, pp. 1–5.

[25] Justin Boyan and Michael Littman. "Packet routing in dynamically changing networks: A reinforcement learning approach." In: *Advances in neural information processing systems* 6 (1993), pp. 671–678.

[26] Steven J Bradtke and Andrew G Barto. "Linear least-squares algorithms for temporal difference learning." In: *Machine learning* 22.1 (1996), pp. 33–57.

[27] GD Caro and Dorigo M Anet. "a Mobile Agents Approach to Adaptive Routing." In: *IRIDA–Universite Libre de Brusseles.—Brussels, Belgium. Technical Report IRIDA* (1998), pp. 1–27.

[28] David Chetret, Chen-Khong Tham, and Lawrence Wai-Choong Wong. "Reinforcement learning and CMAC-based adaptive routing for manets." In: *Proceedings. 2004 12th IEEE International Conference on Networks (ICON 2004)(IEEE Cat. No. 04EX955)*. Vol. 2. IEEE. 2004, pp. 540–544.

[29] Samuel Choi and Dit-Yan Yeung. "Predictive Q-routing: A memory-based reinforcement learning approach to adaptive traffic control." In: *Advances in Neural Information Processing Systems* 8 (1995), pp. 945–951.

[30] Young-ri Choi, Mohaamed G Gouda, Hongwei Zwang, and Anish Arora. *Routing on a logical grid in sensor networks*. Citeseer, 2004, pp. 1–28.

[31] Sultan Mahmood Chowdhury and Ashraf Hossain. "Different energy saving schemes in wireless sensor networks: A survey." In: *Wireless Personal Communications* 114.3 (2020), pp. 2043–2062.

[32] Bin Dai, Yuanyuan Cao, Zhongli Wu, Zhewei Dai, Ruyi Yao, and Yang Xu. "Routing optimization meets Machine Intelligence: A perspective for the future network." In: *Neurocomputing* 459 (2021), pp. 44–58.

[33] Kale Navnath Dattatraya and K Raghava Rao. "Hybrid-based cluster head selection for maximizing network lifetime and energy efficiency in WSN." In: *Journal of King Saud University-Computer and Information Sciences* 34.3 (2022), pp. 716–726.

[34] Bazyli Debowski, Petros Spachos, and Shawki Areibi. "Q-learning enhanced gradient based routing for balancing energy consumption in WSNs." In: *2016 IEEE 21st International Workshop on Computer Aided Modelling and Design of Communication Links and Networks (CAMAD)*. IEEE. 2016, pp. 18–23.

[35] Maarten Devillé, Yann-Aël Le Borgne, Ann Nowé, Patrick De Causmaecker, Joris Maervoet, Tommy Messelis, Katja Verbeeck, and Tim Vermeulen. "Reinforcement learning for energy efficient routing in wireless sensor networks." In: *Proceedings of the Benelux Conference on Artificial Intelligence*. Vol. 23. 2011, pp. 89–96.

[36] Gianni Di Caro and Marco Dorigo. "AntNet: Distributed stigmergetic control for communications networks." In: *Journal of Artificial Intelligence Research* 9 (1998), pp. 317–365.

[37] Edsger W Dijkstra. "A note on two problems in connexion with graphs." In: *Edsger Wybe Dijkstra: His Life, Work, and Legacy*. 2022, pp. 287–290.

[38] Niannian Ding and Peter Xiaoping Liu. "Data gathering communication in wireless sensor networks using ant colony optimization." In: *2004 IEEE International Conference on Robotics and Biomimetics*. IEEE. 2004, pp. 822–827.

[39] Qianao Ding, Rongbo Zhu, Hao Liu, and Maode Ma. "An overview of machine learning-based energy-efficient routing algorithms in wireless sensor networks." In: *Electronics* 10.13 (2021), pp. 1–24.

[40] Asside Christian Djedouboum, Ado Adamou Abba Ari, Abdelhak Mourad Gueroui, Alidou Mohamadou, and Zibouda Aliouat. "Big data collection in large-scale wireless sensor networks." In: *Sensors* 18.12 (2018), pp. 1–34.

[41] Shaoqiang Dong, Prathima Agrawal, and Krishna Sivalingam. "Reinforcement learning based geographic routing protocol for UWB wireless sensor network." In: *IEEE GLOBECOM 2007-IEEE Global Telecommunications Conference*. IEEE. 2007, pp. 652–656.

[42] Praveen Kumar Donta, Tarachand Amgoth, and Chandra Sekhara Rao Annavarapu. "Delay-aware data fusion in duty-cycled wireless sensor networks: A Q-learning approach." In: *Sustainable Computing: Informatics and Systems* 33 (2022), pp. 1–15.

[43] Harsh Doshi and Achyut Shankar. "Wireless Sensor Network Application for IoT-Based Healthcare System." In: *Data Driven Approach Towards Disruptive Technologies: Proceedings of MIDAS 2020*. Springer. 2021, pp. 287–307.

[44] Jason Eisner. "State-of-the-art algorithms for minimum spanning trees-a tutorial discussion." In: *University of Pennsylvania* (1997), pp. 10–21.

[45] Felicia Engmann, Ferdinand Apietu Katsriku, Jamal-Deen Abdulai, Kofi Sarpong Adu-Manu, and Frank Kataka Banaseka. "Prolonging the lifetime of wireless sensor networks: a review of current techniques." In: *Wireless Communications and Mobile Computing* 2018 (2018), pp. 1–23.

[46] Hossam Mahmoud Ahmad Fahmy. *Concepts, applications, experimentation and analysis of wireless sensor networks*. Springer Nature, 2020.

[47] Hossam Mahmoud Ahmad Fahmy. *WSNs applications*. Springer, 2023, pp. 67–242.

[48] Anna Förster and Amy L Murphy. "Balancing energy expenditure in wsns through reinforcement learning: A study." In: *Proceedings of the 1st International Workshop on Energy in Wireless Sensor Networks (WEWSN), Santorini Island, Greece*. Citeseer. 2008, pp. 1–7.

[49] Peng Fu, Jingtao Li, and Deyun Zhang. "Heuristic and distributed QoS route discovery for mobile ad hoc networks." In: *The Fifth International Conference on Computer and Information Technology (CIT'05)*. IEEE. 2005, pp. 512–516.

[50] Reza GhasemAghaei, Md Abdur Rahman, Wail Gueaieb, and Abdulmotaleb El Saddik. "Ant colony-based reinforcement learning algorithm for routing in wireless sensor networks." In: *2007 IEEE instrumentation & measurement technology conference IMTC 2007*. IEEE. 2007, pp. 1–6.

[51] Omprakash Gnawali, Rodrigo Fonseca, Kyle Jamieson, David Moss, and Philip Levis. "Collection tree protocol." In: *Proceedings of the 7th ACM conference on embedded networked sensor systems*. 2009, pp. 1–14.

[52] Pratik Goswami, Ziwei Yan, Amrit Mukherjee, Lixia Yang, Sidheswar Routray, and G Palai. "An energy-efficient clustering using firefly and HML for optical wireless sensor network." In: *Optik* 182 (2019), pp. 181–185.

[53] Gunjan. "A Review on Multi-objective Optimization in Wireless Sensor Networks Using Nature Inspired Meta-heuristic Algorithms." In: *NEURAL PROCESSING LETTERS* (2022), pp. 1–25.

[54] Wen Jing Guo, Cai Rong Yan, Yang Lan Gan, and Ting Lu. "An intelligent routing algorithm in wireless sensor networks based on reinforcement learning." In: *Applied Mechanics and Materials*. Vol. 678. Trans Tech Publ. 2014, pp. 487–493.

[55] Wenjing Guo, Cairong Yan, and Ting Lu. "Optimizing the lifetime of wireless sensor networks via reinforcement-learning-based routing." In: *International Journal of Distributed Sensor Networks* 15.2 (2019), pp. 1–20.

[56] Md Arafat Habib and Sangman Moh. "Robust evolutionary-game-based routing for wireless multimedia sensor networks." In: *Sensors* 19.16 (2019), pp. 1–27.

[57] Aric Hagberg, Pieter Swart, and Daniel S Chult. *Exploring network structure, dynamics, and function using NetworkX*. Tech. rep. Los Alamos National Lab. (LANL), Los Alamos, NM (United States), 2008, pp. 1–6.

[58] Germaine Halegoua. *Smart cities*. MIT press, 2020.

[59] Zahid Halim et al. "Optimizing the minimum spanning tree-based extracted clusters using evolution strategy." In: *Cluster Computing* 21.1 (2018), pp. 377–391.

[60] V Haritha, Swetha Reddy, N Divya Bharathi, Shrikant Upadhyay, R Venkatesh, et al. "Energy Efficient Data Management in Health Care." In: *2023 Third International Conference on Artificial Intelligence and Smart Energy (ICAIS)*. IEEE. 2023, pp. 682–688.

[61] Ahmed Hassan, Ahmed Anter, and Mohammed Kayed. "A robust clustering approach for extending the lifetime of wireless sensor networks in an optimized manner with a novel fitness function." In: *Sustainable Computing: Informatics and Systems* 30 (2021), pp. 1–10.

[62] Tian He, Sudha Krishnamurthy, John A Stankovic, Tarek Abdelzaher, Liqian Luo, Radu Stoleru, Ting Yan, Lin Gu, Jonathan Hui, and Bruce Krogh. "Energy-efficient surveillance system using wireless sensor networks." In: *Proceedings of the 2nd international conference on Mobile systems, applications, and services*. 2004, pp. 270–283.

[63] Wendi Rabiner Heinzelman, Anantha Chandrakasan, and Hari Balakrishnan. "Energy-efficient communication protocol for wireless microsensor networks." In: *Proceedings of the 33rd annual Hawaii international conference on system sciences*. IEEE. 2000, pp. 1–10.

[64] Tiansi Hu and Yunsi Fei. "QELAR: a q-learning-based energy-efficient and lifetime-aware routing protocol for underwater sensor networks." In: *2008 IEEE international performance, computing and communications conference*. IEEE. 2008, pp. 247–255.

[65] Zhang Huanan, Xing Suping, and Wang Jiannan. "Security and application of wireless sensor network." In: *Procedia Computer Science* 183 (2021), pp. 486–492.

[66] Yuanjiang Huang, José-Fernán Martínez, Vicente Hernández Díaz, and Juana Sendra. "A novel topology control approach to maintain the node degree in dynamic wireless sensor networks." In: *Sensors* 14.3 (2014), pp. 4672–4688.

[67] Dheyab Salman Ibrahim, Abdullah Farhan Mahdi, and Qahtan M Yas. "Challenges and issues for wireless sensor networks: a survey." In: *Journal of global scientific research* 6.1 (2021), pp. 1079–1097.

[68] Dheyab Salman Ibrahim, Fadhil Kadhem Zaidan, et al. "Routing protocols for energy efficiency in WSNs: a review." In: *SAR Journal* 4.1 (2021), pp. 29–33.

[69] Savio D Immanuel and Udit Kr Chakraborty. "Genetic algorithm: an approach on optimization." In: *2019 international conference on communication and electronics systems (ICCES)*. IEEE. 2019, pp. 701–708.

[70] Chalermek Intanagonwiwat, Ramesh Govindan, and Deborah Estrin. "Directed diffusion: A scalable and robust communication paradigm for sensor networks." In: *Proceedings of the 6th annual international conference on Mobile computing and networking*. 2000, pp. 56–67.

[71] Chalermek Intanagonwiwat, Ramesh Govindan, Deborah Estrin, John Heidemann, and Fabio Silva. "Directed diffusion for wireless sensor networking." In: *IEEE/ACM transactions on networking* 11.1 (2003), pp. 2–16.

[72] Sohail Jabbar, Muhammad Asif Habib, Abid Ali Minhas, Mudassar Ahmad, Rehan Ashraf, Shehzad Khalid, and Kijun Han. "Analysis of factors affecting energy-aware routing in wireless sensor network." In: *Wireless Communications and Mobile Computing* 2018 (2018), pp. 1–21.

[73] Sara Zafar Jafarzadeh and Mohammad Hossein Yaghmaee Moghaddam. "Design of energy-aware QoS routing protocol in wireless sensor networks using reinforcement learning." In: *2014 IEEE 27th Canadian Conference on Electrical and Computer Engineering (CCECE)*. IEEE. 2014, pp. 1–5.

[74] Peggy James, Ross Astoria, Theresa Castor, Christopher Hudspeth, Denise Olstinske, and John Ward. "Smart cities: Fundamental concepts." In: *Handbook of smart cities* (2021), pp. 3–33.

[75] Upasna Joshi and Rajiv Kumar. "Reinforcement learning based energy efficient protocol for wireless multimedia sensor networks." In: *Multimedia Tools and Applications* 81.2 (2022), pp. 2827–2840.

[76] Leslie Pack Kaelbling, Michael L Littman, and Andrew W Moore. "Reinforcement learning: A survey." In: *Journal of artificial intelligence research* 4 (1996), pp. 237–285.

[77] Dionisis Kandris, Christos Nakas, Dimitrios Vomvas, and Grigorios Koulouras. "Applications of wireless sensor networks: an up-to-date survey." In: *Applied System Innovation* 3.1 (2020), pp. 1–24.

[78] Brad Karp and Hsiang-Tsung Kung. "GPSR: Greedy perimeter stateless routing for wireless networks." In: *Proceedings of the 6th annual international conference on Mobile computing and networking*. 2000, pp. 243–254.

[79] B Karthikeyan, K Alhaf Malik, D Bujji Babbu, K Nithya, S Jafar Ali Ibrahim, and NS Kalyan Chakravarthy. "Survey of Cooperative Routing Algorithms in Wireless Sensor Networks." In: *Annals of the Romanian Society for Cell Biology* (2021), pp. 5316–5320.

[80] Ramgopal Kashyap. "Applications of wireless sensor networks in healthcare." In: *IoT and WSN applications for modern agricultural advancements: Emerging research and opportunities*. IGI Global, 2020, pp. 8–40.

[81] Amar Kaswan, Kumar Nitesh, and Prasanta K Jana. "Energy efficient path selection for mobile sink and data gathering in wireless sensor networks." In: *AEU-International Journal of Electronics and Communications* 73 (2017), pp. 110–118.

[82] Sourabh Katoch, Sumit Singh Chauhan, and Vijay Kumar. "A review on genetic algorithm: past, present, and future." In: *Multimedia Tools and Applications* 80 (2021), pp. 8091–8126.

[83] Navpreet Kaur and Inderdeep Kaur Aulakh. "An Energy Efficient Reinforcement Learning Based Clustering Approach for Wireless Sensor Network." In: *EAI Endorsed Transactions on Scalable Information Systems* 8.31 (2021), pp. 1–17.

[84] Hee-won Kim, Junho Cho, and Ho-Shin Cho. "Topology-Aware Reinforcement Learning Routing Protocol in Underwater Wireless Sensor Networks." In: *2019 International Conference on Information and Communication Technology Convergence (ICTC)*. IEEE. 2019, pp. 124–126.

[85] Hlabishi I Kobo, Adnan M Abu-Mahfouz, and Gerhard P Hancke. "A survey on software-defined wireless sensor networks: Challenges and design requirements." In: *IEEE access* 5 (2017), pp. 1872–1899.

[86] Oliver Kramer and Oliver Kramer. *Genetic algorithms*. Springer, 2017.

[87] Joseph B Kruskal. "On the shortest spanning subtree of a graph and the traveling salesman problem." In: *Proceedings of the American Mathematical society* 7.1 (1956), pp. 48–50.

[88] S Kumar and R Miikkulainen. "Confidence-based Q-routing: An online network routing algorithm." In: *Proc. in 16th Conf. Artif. Neural Netw. Eng.* 1998, pp. 758–763.

[89] Michail G Lagoudakis and Ronald Parr. "Least-squares policy iteration." In: *The Journal of Machine Learning Research* 4 (2003), pp. 1107–1149.

[90] Annu Lambora, Kunal Gupta, and Kriti Chopra. "Genetic algorithm-A literature review." In: *2019 international conference on machine learning, big data, cloud and parallel computing (COMITCon)*. IEEE. 2019, pp. 380–384.

[91] Anna Lanzolla and Maurizio Spadavecchia. "Wireless sensor networks for environmental monitoring." In: *Sensors* 21.4 (2021), pp. 1–3.

[92] TT Le and Sangman Moh. "Reinforcement-learning-based topology control for wireless sensor networks." In: *Proceedings of the Grid and Distributed Computing* 2016 (2016), pp. 22–7.

[93] Chun-Ta Li. "Security of wireless sensor networks: current status and key issues." In: *Smart Wireless Sensor Networks* (2010), pp. 299–313.

[94] Yuanzhen Li, Yang Zhao, and Yingyu Zhang. "A spanning tree construction algorithm for industrial wireless sensor networks based on quantum artificial bee colony." In: *EURASIP Journal on Wireless Communications and Networking* 2019.1 (2019), pp. 1–12.

[95] Yong Lu, Guangzhou Zhao, and Fanjun Su. "Adaptive ant-based dynamic routing algorithm." In: *Fifth World Congress on Intelligent Control and Automation (IEEE Cat. No. 04EX788)*. Vol. 3. IEEE. 2004, pp. 2694–2697.

[96] Haiyun Luo, Fan Ye, Jerry Cheng, Songwu Lu, and Lixia Zhang. "TTDD: Two-tier data dissemination in large-scale wireless sensor networks." In: *Wireless networks* 11 (2005), pp. 161–175.

[97] Naila Nawaz Malik, Wael Alosaimi, M Irfan Uddin, Bader Alouffi, and Hashem Alyami. "Wireless sensor network applications in healthcare and precision agriculture." In: *Journal of Healthcare Engineering* 2020 (2020), pp. 1–9.

[98] Zoubir Mammeri. "Reinforcement learning based routing in networks: Review and classification of approaches." In: *Ieee Access* 7 (2019), pp. 55916–55950.

[99] Miklos Maroti. "Directed flood-routing framework for wireless sensor networks." In: *ACM/IFIP/USENIX International Conference on Distributed Systems Platforms and Open Distributed Processing*. Springer. 2004, pp. 99–114.

[100] Mohammad Abdul Matin and MM Islam. "Overview of wireless sensor network." In: *Wireless sensor networks-technology and protocols* 1.3 (2012), pp. 3–24.

[101] Kenji Miyao, Hidehisa Nakayama, Nirwan Ansari, and Nei Kato. "LTRT: An efficient and reliable topology control algorithm for ad-hoc networks." In: *IEEE Transactions on Wireless Communications* 8.12 (2009), pp. 6050–6058.

[102] Habib Mostafaei and Michael Menth. "Software-defined wireless sensor networks: A survey." In: *Journal of Network and Computer Applications* 119 (2018), pp. 42–56.

[103] Vially Kazadi Mutombo, Sung Y Shin, and Jiman Hong. "EBR-RL: energy balancing routing protocol based on reinforcement learning for WSN." In: *Proceedings of the 36th Annual ACM Symposium on Applied Computing*. 2021, pp. 1915–1920.

[104] Christos Nakas, Dionisis Kandris, and Georgios Visvardis. "Energy efficient routing in wireless sensor networks: A comprehensive survey." In: *Algorithms* 13.3 (2020), pp. 1–65.

[105] Wibhada Naruephiphat and Wipawee Usaha. "Balancing tradeoffs for energy-efficient routing in MANETs based on reinforcement learning." In: *VTC Spring 2008-IEEE Vehicular Technology Conference*. IEEE. 2008, pp. 2361–2365.

[106] Padmalaya Nayak, GK Swetha, Surbhi Gupta, and K Madhavi. "Routing in wireless sensor networks using machine learning techniques: Challenges and opportunities." In: *Measurement* 178.108974 (2021), pp. 1–15.

[107] Petteri Nurmi. "Reinforcement learning for routing in ad hoc networks." In: *2007 5th international symposium on modeling and optimization in mobile, ad hoc and wireless networks and workshops*. IEEE. 2007, pp. 1–8.

[108] Guido Oddi, Antonio Pietrabissa, and Francesco Liberati. "Energy balancing in multi-hop Wireless Sensor Networks: an approach based on reinforcement learning." In: *2014 NASA/ESA Conference on Adaptive Hardware and Systems (AHS)*. IEEE. 2014, pp. 262–269.

[109] Oladayo Olufemi Olakanmi and Adedamola Dada. "Wireless sensor networks (WSNs): Security and privacy issues and solutions." In: *Wireless mesh networks-security, architectures and protocols* (2020), pp. 13–30.

[110] AB Patel and Hitesh B Shah. "Reinforcement learning framework for energy efficient wireless sensor networks." In: *International Research Journal of Engineering and Technology IRJET* 2 (2015), pp. 128–134.

[111] Leonid Peshkin and Virginia Savova. "Reinforcement learning for adaptive routing." In: *Proceedings of the 2002 International Joint Conference on Neural Networks. IJCNN'02 (Cat. No. 02CH37290)*. Vol. 2. IEEE. 2002, pp. 1825–1830.

[112] Srinivasa Prasanna and Srinivasa Rao. "An overview of wireless sensor networks applications and security." In: *International Journal of Soft Computing and Engineering* 2.2 (2012), pp. 2231–2307.

[113] Robert Clay Prim. "Shortest connection networks and some generalizations." In: *The Bell System Technical Journal* 36.6 (1957), pp. 1389–1401.

[114] Rahul Priyadarshi, Bharat Gupta, and Amulya Anurag. "Deployment techniques in wireless sensor networks: a survey, classification, challenges, and future research issues." In: *The Journal of Supercomputing* 76.9 (2020), pp. 7333–7373.

[115] Meena Pundir and Jasminder Kaur Sandhu. "A systematic review of quality of service in wireless sensor networks using machine learning: Recent trend and future vision." In: *Journal of Network and Computer Applications* 188 (2021), pp. 1–33.

[116] G Mohan Ram and E Ilavarsan. "Review on energy-efficient routing protocols in WSN." In: *Computer Networks, Big Data and IoT: Proceedings of ICCBI 2020* (2021), pp. 851–871.

[117] SR Jino Ramson and D Jackuline Moni. "Applications of wireless sensor networks—A survey." In: *2017 international conference on innovations in electrical, electronics, instrumentation and media technology (ICEEIMT)*. IEEE. 2017, pp. 325–329.

[118] Priyanka Rawat, Kamal Deep Singh, Hakima Chaouchi, and Jean Marie Bonnin. "Wireless sensor networks: a survey on recent developments and potential synergies." In: *The Journal of supercomputing* 68.1 (2014), pp. 1–48.

[119] A Pravin Renold and S Chandrakala. "MRL-SCSO: multi-agent reinforcement learning-based self-configuration and self-optimization protocol for unattended wireless sensor networks." In: *Wireless Personal Communications* 96.4 (2017), pp. 5061–5079.

[120] Justus Rischke, Peter Sossalla, Hani Salah, Frank HP Fitzek, and Martin Reisslein. "QR-SDN: Towards Reinforcement Learning States, Actions, and Rewards for Direct Flow Routing in Software-Defined Networks." In: *IEEE Access* 8 (2020), pp. 174773–174791.

[121] Tirtharaj Sapkota and Bobby Sharma. "Analyzing the energy efficient path in Wireless Sensor Network using Machine Learning." In: *ADBU Journal of Engineering Technology* 10 (2021), pp. 1–7.

[122] Shivangi Satija, Tejsi Sharma, and Bharat Bhushan. "Innovative approach to wireless sensor networks: Sd-wsn." In: *2019 International Conference on Computing, Communication, and Intelligent Systems (ICCCIS)*. IEEE. 2019, pp. 170–175.

[123] T Shankar, S Shanmugavel, and A Rajesh. "Hybrid HSA and PSO algorithm for energy efficient cluster head selection in wireless sensor networks." In: *Swarm and Evolutionary Computation* 30 (2016), pp. 1–10.

[124] Varun K Sharma, Shiv Shankar Prasad Shukla, and Varun Singh. "A tailored Q-Learning for routing in wireless sensor networks." In: *2012 2nd IEEE International Conference on Parallel, Distributed, and Grid Computing*. IEEE. 2012, pp. 663–668.

[125] Abhilash Singh, Sandeep Sharma, and Jitendra Singh. "Nature-inspired algorithms for wireless sensor networks: A comprehensive survey." In: *Computer Science Review* 39 (2021), pp. 1–23.

[126] Adam Slowik and Halina Kwasnicka. "Evolutionary algorithms and their applications to engineering problems." In: *Neural Computing and Applications* 32 (2020), pp. 12363–12379.

[127] Georgios Smaragdakis, Ibrahim Matta, Azer Bestavros, et al. "SEP: A stable election protocol for clustered heterogeneous wireless sensor networks." In: *Second international workshop on sensor and actor network protocols and applications (SANPA 2004)*. Vol. 3, pp. 1–11.

[128] Kazem Sohraby, Daniel Minoli, and Taieb Znati. *Wireless sensor networks: technology, protocols, and applications*. John wiley & sons, 2007.

[129] Santosh Soni and Manish Shrivastava. "Novel learning algorithms for efficient mobile sink data collection using reinforcement learning in wireless sensor network." In: *Wireless Communications and Mobile Computing* 2018 (2018), pp. 1–13.

[130] Devika Subramanian, Peter Druschel, and Johnny Chen. "Ants and reinforcement learning: A case study in routing in dynamic networks." In: *IJCAI (2)*. Citeseer. 1997, pp. 832–839.

[131] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[132] Md Shahid Thekiya and Mangesh D Nikose. "Role of Routing Techniques in Wireless Sensor Networks–A Survey." In: *Distributed Computing and Optimization Techniques: Select Proceedings of ICDCOT 2021* (2022), pp. 761–770.

[133] Chai K Toh, Julio A Sanguesa, Juan C Cano, and Francisco J Martinez. "Advances in smart roads for future smart cities." In: *Proceedings of the Royal Society A* 476.2233 (2020), pp. 1–24.

[134] Jia Uddin. "A Novel Data Aggregation Mechanism using Reinforcement Learning for Cluster Heads in Wireless Multimedia Sensor Networks." In: *Annals of Emerging Technologies in Computing (AETiC)* 6.3 (2022), pp. 69–77.

[135] Kingsley Eghonghon Ukhurebor, Ituabhor Odesanya, Silas Soo Tyokighir, Rout George Kerry, Akinola Samson Olayinka, and Ayodotun Oluwafemi Bobadoye. "Wireless Sensor Networks: Applications and Challenges." In: *Wireless Sensor Networks-Design, Deployment and Applications* (2021), pp. 25–37.

[136] Jin Wang, Yiquan Cao, Bin Li, Hye-jin Kim, and Sungyoung Lee. "Particle swarm optimization based clustering algorithm with mobile sink for WSNs." In: *Future Generation Computer Systems* 76 (2017), pp. 452–457.

[137] Ping Wang and Ting Wang. "Adaptive routing for sensor networks using reinforcement learning." In: *The Sixth IEEE International Conference on Computer and Information Technology (CIT'06)*. IEEE. 2006, pp. 219–219.

[138] Tianshu Wang, Gongxuan Zhang, Xichen Yang, and Ahmadreza Vajdi. "Genetic algorithm for energy-efficient clustering and routing in wireless sensor networks." In: *Journal of Systems and Software* 146 (2018), pp. 196–214.

[139] Brett Warneke, Matt Last, Brian Liebowitz, and Kristofer SJ Pister. "Smart dust: Communicating with a cubic-millimeter computer." In: *Computer* 34.1 (2001), pp. 44–51.

[140] Christopher JCH Watkins and Peter Dayan. "Q-learning." In: *Machine learning* 8.3-4 (1992), pp. 279–292.

[141] Yao-feng Wen, Yu-quan Chen, and Min Pan. "Adaptive ant-based routing in wireless sensor networks using Energy and Delay metrics." In: *Journal of Zhejiang University-SCIENCE A* 9.4 (2008), pp. 531–538.

[142] Darrell Whitley. "A genetic algorithm tutorial." In: *Statistics and computing* 4.2 (1994), pp. 65–85.

[143] Feng Xia. *Wireless sensor technologies and applications*. 2009.

[144] Takeo Yamada, Seiji Kataoka, and Kohtaro Watanabe. "Listing all the minimum spanning trees in an undirected graph." In: *International Journal of Computer Mathematics* 87.14 (2010), pp. 3175–3185.

[145] Jun Yang, Hesheng Zhang, Cheng Pan, and Wei Sun. "Learning-based routing approach for direct interactions between wireless sensor network and moving vehicles." In: *16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013)*. IEEE. 2013, pp. 1971–1976.

[146] Kok-Lim Alvin Yau, Hock Guan Goh, David Chieng, and Kae Hsiang Kwong. "Application of reinforcement learning to wireless sensor networks: models and algorithms." In: *Computing* 97 (2015), pp. 1045–1075.

[147] Rachid Zagrouba and Amine Kardi. "Comparative study of energy efficient routing techniques in wireless sensor networks." In: *Information* 12.1 (2021), pp. 1–28.

[148] Ying Zhang and Markus Fromherz. "Constrained flooding: a robust and efficient routing framework for wireless sensor networks." In: *20th International Conference on Advanced Information Networking and Applications-Volume 1 (AINA'06)*. Vol. 1. IEEE. 2006, pp. 1–6.

[149] Ying Zhang and Qingfeng Huang. "A Learning-based Adaptive Routing Tree for Wireless Sensor Networks." In: *J. Commun.* 1.2 (2006), pp. 12–21.

[150] Ying Zhang, Lukas D Kuhn, and Markus PJ Fromherz. "Improvements on ant routing for sensor networks." In: *Lecture notes in computer science* 3172 (2004), pp. 154–165.

[151] Milica Pejanović Đurišić, Zhilbert Tafa, Goran Dimić, and Veljko Milutinović. "A survey of military applications of wireless sensor networks." In: *2012 Mediterranean conference on embedded computing (MECO)*. IEEE. 2012, pp. 196–199.