computers

MDPI

Article

A Centralized Routing for Lifetime and Energy Optimization in WSNs Using Genetic Algorithm and Least-Square Policy Iteration

Elvis Obi [1,*], Zoubir Mammeri [1] and Okechukwu E. Ochia [2]

[1] Computer Science Research Institute, Paul Sabatier University, 31062 Toulouse, France
[2] Department of Electrical and Computer Engineering, University of Calgary, Calgary, AB T2N 1N4, Canada
* Correspondence: elvis.obi@irit.fr; Tel.: +33-7535-80043

Abstract: Q-learning has been primarily used as one of the reinforcement learning (RL) techniques to find the optimal routing path in wireless sensor networks (WSNs). However, for the centralized RL-based routing protocols with a large state space and action space, the baseline Q-learning used to implement these protocols suffers from degradation in the convergence speed, network lifetime, and network energy consumption due to the large number of learning episodes required to learn the optimal routing path. To overcome these limitations, an efficient model-free RL-based technique called Least-Square Policy Iteration (LSPI) is proposed to optimize the network lifetime and energy consumption in WSNs. The resulting designed protocol is a Centralized Routing Protocol for Lifetime and Energy Optimization with a Genetic Algorithm (GA) and LSPI (CRPLEOGALSPI). Simulation results show that the CRPLEOGALSPI has improved performance in network lifetime and energy consumption compared to an existing Centralized Routing Protocol for Lifetime Optimization with GA and Q-learning (CRPLOGARL). This is because the CRPLEOGALSPI chooses a routing path in a given state considering all the possible routing paths, and it is not sensitive to the learning rate. Moreover, while the CRPLOGARL evaluates the optimal policy from the Q-values, the CRPLEOGALSPI updates the Q-values based on the most updated information regarding the network dynamics using weighted functions.

Keywords: wireless sensor network; routing; network lifetime; energy consumption; reinforcement learning; path optimization; least-squares policy iteration

Citation: Obi, E.; Mammeri, Z.; Ochia, O.E. A Centralized Routing for Lifetime and Energy Optimization in WSNs Using Genetic Algorithm and Least-Square Policy Iteration. *Computers* **2023**, *12*, 22. https://doi.org/10.3390/computers12020022

Academic Editor: Paolo Bellavista

Received: 16 December 2022
Revised: 5 January 2023
Accepted: 11 January 2023
Published: 18 January 2023

1. Introduction

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 a sink using the wireless medium [1].

The advantages of WSN technology, when compared to traditional solutions of networking, are scalability, low costs, accuracy, reliability, flexibility, and deployment ease [2]. 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 the limited processing and computing speed of the sensor nodes [3].

Basically, 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 [4]. 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. 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 [5].

Thus, routing in WSNs is an energy-consuming technique, which makes energy consumption and increasing network lifetime major challenges in WSNs [6]. 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.

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. 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 an NP-hard problem.

This makes routing that uses traditional route optimization techniques based on a deterministic algorithm or Dynamic Programming (DP) 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. 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 [7].

The energy-efficient utilization of sensor nodes can be achieved using three control techniques, which are decentralized control, distributed control, and centralized control [8]. 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. For distributed control, each node makeslocal decisions with its partial knowledge of the entire network. This normally results in non-optimal routes in terms of energy consumption. By contrast, in 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.

The global knowledge of the sink 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. 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. The possible routing tables of the sensor nodes can be the possible Minimum Spanning Trees (MSTs) generated by the sink after the network initialization [9].

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. 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.

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.

RL is a category of machine learning that solves a problem by learning with the trial-and-error method [10]. The Lifetime-Aware Centralized Q-routing Protocol (LACQRP) [9] uses an All-MSTs algorithm [11] 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 LACQRP unfeasible in practice for very large WSNs. GA, a type of evolutionary and search-based adaptive heuristic algorithm [12], 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 Protocol for Lifetime Optimization using Genetic Algorithm and Reinforcement Learning (CRPLOGARL) [13] enables the sink to generate a subset of MSTs for a large-scale WSN graph in polynomial time. The subset MSTs are then used as routing tables by the sensor nodes to send data packets to the sink. The LACQRP and the CRPLOGARL use Q-learning [14] to learn the routing tables that maximize the lifetime of the WSNs.

However, due to the large state space and action space of these protocols, the baseline Q-learning used to implement these protocols suffers 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) [15] is used to replace Q-learning, because LSPI chooses a routing path in a given state considering all possible routing paths, and it 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. Therefore, this paper presents the design of a Centralized Routing Protocol for Lifetime and Energy Optimization using GA and LSPI (CRPLEOGALSPI) for WSNs.

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 a GA. 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 paper 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.

The remaining of the paper is presented as follows: Section 2 is the literature review. Section 3 explains the design of the CRPLEOGALSPI. Section 4 is the simulation and discussion of results, and Section 5 is the conclusion.

2. Literature Review

2.1. Fundamental Concepts

RL is a type of machine learning that makes an agent learn the dynamic behavior of its environment by taking actions based on its current state, which improves learning with time (maximizing the cumulative reward concept) using trial-and-error interaction with the environment [10]. 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 sink, the environment is the sink's neighborhood, the state is the current unicast 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 data packets. An RL problem is solved by modeling the problem as a Markov Decision Process (MDP), as shown in Figure 1. The MDP is made up of four tuples (S, A, P, R), where S denotes the states set, A is the actions set, P is the state transition probability matrix, and R is the reward function. P and R combined together are used to describe the model of the environment. The probability of being in a given state $S_{t+1} = \hat{s}$ from a current state $S_t = s$ by taking action $A_t = a$, and getting a reward $R_{t+1} = r$ is known as the transitional probability, given in Equation (1) [16].

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

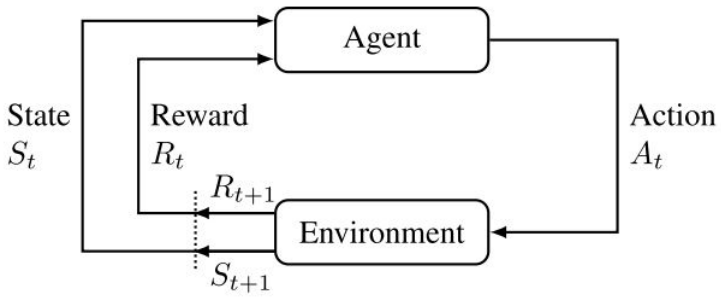

Figure 1. RL Model as an MDP [10].

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 generally used in practice due to their dependency on the initial environment model's accuracy and the larger size of storage costs 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 maps states to actions. A policy π_t determines the learning behavior 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, whereas 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 function. The action-value function computes how good it is for an agent in a given state to take a given action. At the same time, 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 [17]. This is because they do not need the network's environment models, which are difficult to get as a result of their dynamically changing properties like the residual battery capacities, lifetime of nodes, etc., in WSNs.

In RL, the quality of taking an action, $A_t = a$ from a state, $S_t = s$, is given using 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 learn an optimal policy that maps states to actions in order to maximize 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 shown in Equation (2).

$$Q^{\pi}(s, a) = r(s, a) + \gamma \sum_{\hat{s}} p(\hat{s}|s, a) \max_{\hat{a}} \{Q^{\pi}(\hat{s}, \hat{a})\} \quad (2)$$

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 obtained immediately by taking an action a when in a state s, and it is calculated as in Equation (3):

$$r(s,a) = \sum_{\hat{s}} p(\hat{s}|s,a)R(s,a,\hat{s}) \tag{3}$$

where $p(\hat{s}|s,a)$ is the probability transition model that gives the probability of being in a state $\hat{s}$, after taking an action a when in state s. $R(s,a,\hat{s})$ is the immediate reward obtained when an agent takes an action a when at state s in transit to state $\hat{s}$. The last term in Equation (2) 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).

2.1.1. 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 are known [14]. 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 the optimum action values with one as the probability if, and only if, all actions are continuously sampled in all of the states. In Q-learning, the quality of selecting action $a \in A$ when in a state $s \in S$ is based on the calculated state–action-value function called Q-value. The Q-values give the measure of accumulated rewards an agent gets from all pairs of state–action. The estimate of the state–action-value function $Q(s,a)$ that an agent used to learn the best action in a particular state is achieved by storing the Q-values $Q(s_t,a_t)$ of pairs of state–action with the iterative update rule in Equation (4).

$$Q(s_t,a_t) = (1-\alpha)Q(s_t,a_t) + \alpha\left[R_{t+1} + \gamma * max\{Q(s_{t+1},a)\}\right] \tag{4}$$

where $0 < \alpha \leq 1$ represent the learning rate and $0 \leq \gamma \leq 1$ represent the discount factor. The learning rate gives the degree to which the later learned Q-value impacts the former Q-value. The nearer the magnitude of α is to one, the more the effect of the later calculated Q-value on the former one. If α is one, then the later computed Q-value replaces the former Q-value totally. The discount factor manages the agent's appreciation for hereafter rewards with respect to the recent reward. If γ is 1, both the recent reward and the hereafter reward are regarded equally.

Q-learning, as a model-free off-policy RL technique, is appropriate for learning optimal routing rules in WSNs because of its simplicity and non-requirement for any knowledge of the underlying transition and reward mechanism. However, Q-learning suffers the following drawbacks if it is used in learning optimal routing rules 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.

2.1.2. 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) [15]. 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, a set consisting of k state–action-dependent basic functions $\varphi(s,a)$ provides the information of the selected state–action pair features as given in Equation (5).

$$\mathcal{F} = \left\{ \varphi_j(s,a) : S \times A \mapsto R,\ j = 1,\ldots,k \right\} \tag{5}$$

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 (6).

$$\hat{Q}^{\pi}(s,a) = \sum_{j=1}^{k} \varphi_j(s,a) w_j = \varphi(s,a)^T w \tag{6}$$

where w_j is the weight associated with the jth basic function. From the approximated form of Equation (2), using Equation (6), the matrix form of Equation (2) can be written as in Equation (7).

$$\Psi w \approx R + \gamma P^{\pi} \Psi w \tag{7}$$

where Ψ is a matrix of basic functions for each state–action pair and is of size $|S||A| \times k$.

Equation (7) can be reformulated as Equation (8) for a linearly dependent column of Ψ.

$$\Psi^T(\Psi - \gamma P^{\pi}\Psi) w^{\pi} = \Psi^T R \tag{8}$$

Solving the linear system in Equation (8) leads to the extraction of the weights associated with $\hat{Q}^{\pi}(s,a)$ in Equation (6). The equation for extracting the weights can be written as in Equation (9).

$$w^{\pi} = X^{-1} y \tag{9}$$

where X and y are given in Equations (10) and (11), respectively.

$$X = \Psi^T(\Psi - \gamma P^{\pi}\Psi) \tag{10}$$

$$y = \Psi^T R \tag{11}$$

LSPI, as 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 Least-Squares Temporal-Difference Learning (LSTDQ) [15]. The LSTDQ is an algorithm similar to the least-squares temporal-difference learning algorithm (LSTD) [18] 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 $\mathcal{D} = \left\{(s_i, a_i, r_i, \hat{s_i}) | i = 1, 2 \ldots, M\right\}$ obtained from the environment, the approximated versions of Ψ, $P^{\pi}\Psi$, and R are constructed using Equations (12)–(14), respectively.

$$\hat{\Psi} = \begin{pmatrix} \Psi(s_1, a_1)^T \\ \vdots \\ \Psi(s_i, a_i)^T \\ \vdots \\ \Psi(s_M, a_M)^T \end{pmatrix} \tag{12}$$

$$P^{\hat{\pi}}\Psi = \begin{pmatrix} \Psi(\hat{s_1}, \pi(\hat{s_1}))^T \\ \vdots \\ \Psi(\hat{s_i}, \pi(\hat{s_i}))^T \\ \vdots \\ \Psi(\hat{s_M}, \pi(\hat{s_M}))^T \end{pmatrix} \tag{13}$$

$$\hat{R} = \begin{pmatrix} r_1 \\ \vdots \\ r_i \\ \vdots \\ r_M \end{pmatrix} \tag{14}$$

Therefore, the approximated $\hat{X}$ and $\hat{y}$ can be given as in Equations (15) and (16), respectively.

$$\hat{X} = \frac{\hat{\Psi^T}(\hat{\Psi} - \gamma P^{\hat{\pi}}\Psi)}{M} \tag{15}$$

$$\hat{y} = \frac{\hat{\Psi^T}\hat{R}}{M} \tag{16}$$

Since in a practical evaluation of $\hat{X}$ and $\hat{y}$, M is finite, 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 obtained 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 $\hat{X}^i$ and $\hat{y}^i$, respectively. Therefore, Equations (17) and (18) will give the approximated values of $\hat{X}^{i+1}$ and $\hat{y}^{i+1}$, respectively for a new sample $(s_i, a_i, r_i, \hat{s_i})$.

$$\hat{X}^{i+1} = \hat{X}^i + \varphi(s_i, a_i)\left[\varphi(s_i, a_i) - \gamma\varphi(\hat{s_i}, \pi(\hat{s_i}))\right]^T \tag{17}$$

$$\hat{y}^{i+1} = \hat{y}^i + \varphi(s_i, a_i)r_i \tag{18}$$

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 (19).

$$\pi(s|w_\pi) = \underset{a}{\text{argmax}}(\varphi(s, a)^T w_\pi) \tag{19}$$

2.2. Review of Similar Works

The first hop-by-hop routing protocol to utilize RL is called Q-routing, proposed by Boyan and Littman [19]. Q-routing minimizes the packet delivery delay. However, Q-routing is limited by drawbacks that include Q-value freshness, sensitivity to parameter settings, and slow convergence to the optimal routing rules. Different works have considered the design of RL routing protocols to minimize energy consumption or/and maximize the lifetime of WSNs. These works are presented in the sequel.

Zhang and Fromherz [20] designed RL-based constrained flooding for WSNs to optimize the number of packets transmitted when sending data packets from source nodes to the sink. The cost of flooding was 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 RL-based constrained flooding action is packet broadcasting using either constrained propagation, differential delay, or probabilistic retransmission without using control packets. The RL-based constrained flooding has improved energy efficiency when compared with direct routing [21] and a backbone tree [22] that uses direct diffusion [23] with simulations. However, direct routing has a better packet delivery delay than RL-based constrained flooding.

Wang and Wang [24] proposed a routing algorithm called Adaptive Routing for WSNs, using RL (AdaR) to maximize the network lifetime. The protocol uses multiple factors including 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.

Nurmi (2007) proposed an energy-aware and selfish RL-based routing protocol for ad hoc networks [25]. 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 were not provided since the protocol was generic.

Dong et al. [26] proposed a Reinforcement Learning Based Geographical Routing Protocol (RLGR) for ultra-wideband sensor networks. The protocol seeks to improve the network lifetime by reducing packet delivery delay and distributing energy consumption among nodes uniformly. The RLGR considers hop counts to the sink and residual energy of nodes when choosing the next forwarder. The RLGR improves the network lifetime by at least 75 percent when compared with Greedy Perimeter Stateless Routing (GPSR) in a simulation [27].

Arroyo-Valles et al. [28] proposed a geographical routing algorithm for WSNs called Q-Probabilistic routing (Q-PR). 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.

Naruephiphat and Usaha [29] proposed a routing protocol for a Mobile ad hoc network (MAGNET) to balance the tradeoff between minimizing energy consumption and maximizing network lifetime. This is achieved by using RL to select routing paths based on the energy consumption of paths and 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.

Forster and Murphy [30] designed a distributed multicast routing algorithm based on RL called E-FROMS. E-FROMS balanced the energy consumption in a multiple-sink 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 are high and very high, respectively.

Hu and Fei [31] proposed a Q-learning-based energy-efficient and lifetime-aware routing (QELAR) for an underwater sensor network (UWSN) to find the optimal routing path from each source sensor node to the sink. QELAR increases the lifetime of the network by distributing the residual energy of each sensor node evenly. This is achieved by each sensor node learning the optimal routing path using a reward function that considers the sensor node's residual energy and a group of sensor nodes' distribution energy while forwarding data packets by each sensor node.

Yang et al. [32] proposed a reinforcement learning-based routing protocol between sensor nodes and mobile sinks, which are vehicles. 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.

Oddi et al. [33] modify the Q-Routing protocol, designed for wired networks, to enable the application for WSNs. The proposed routing protocol, called optimized Q-Routing (OPT-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.

Jafarzadeh and Moghaddam [34] proposed a routing protocol for WSNs called Energy-aware QoS routing RL-based (EQR-RL). EQR-RL minimizes the energy consumption in WSNs while ensuring the packet delivery delay. EQR-RL selects the next forwarder of data packets using the probability distribution-based exploration strategy. The reward function considered in designing EQR-RL is the combination of the weighted metrics of the residual energy of the selected forwarder, the ratio of packets between the packet sender to the selected forwarder, and link delay.

Guo et al. [35] designed an intelligent routing protocol for WSNs built on RL named the reinforcement-learning-based lifetime optimization (RLLO) routing protocol. The reward function used by RLLO to update the Q-values of the agents is the sensor node's residual energy and the hop counts of the sensor nodes to the sink. The performance analysis of RLLO is carried out with simulations and the result showed that RLLO had improved network lifetime and packet delivery delay when compared with energy-aware routing (EAR) [36] and improved energy-aware routing (I-EAR) [37].

Debowski et al. [38] proposed a hybrid protocol called Q-Smart Gradient-based routing protocol (QSGrd). QSGrd minimizes the energy consumed by the sensor nodes in the WSNs by jointly using Q-learning and a transmission gradient. In QSGrd, each sensor node neighbor is associated with successful transmission probability, which is a function of the distance between nodes and maximum transmission range. The success transmission probabilities of the sensor node's neighbors cause a transmission gradient. Subsequently, the Q-values are updated by the success transmission probabilities. The optimal routing paths are calculated with the mean smaller number of packet transmissions to the destination, which is the sink and the residual energy of the next hop with Q-learning.

Renold & Chandrakala [39] proposed a routing protocol for WSNs called Multi-agent Reinforcement Learning-based Self-Configuration and Self-Optimization (MRL-SCSO). 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) [40] when compared using a simulation.

Geo et al. [41] proposed a Q-learning routing protocol for WSNs called Reinforcement Learning-Based Routing (RLBR) to optimize the network lifetime. RLBR searches 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 a transmit power adjusting and data packet carrying feedback scheme to increase packet delivery, balance energy consumption, and reduce overall energy consumption. RLBR performs better than EAR, Balanced energy-efficient routing (BEER) [37], Q-Routing, and MRL-SCSO in terms of network lifetime and energy efficiency.

Bouzid et al. [42] proposed a routing protocol to optimize lifetime and energy consumption. 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 using a simulation, and the results show that R2LTO performs better in terms of network lifetime and energy efficiency.

Sapkota and Sharma [43] designed an RL-based routing protocol to optimize the network lifetime in WSN. 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 had improved performance when compared with the baseline direct diffusion protocol [44].

Mutombo et al. [45] proposed an RL-based Energy Balancing Routing (EBR-RL) protocol for WSNs. EBR-RL protocol balances the energy consumed between sensor nodes and thereby maximizes the lifetime of the WSN. The EBR-RL protocol works in two phases. The first phase is the network setup and initialization, while the second phase is learning the optimal path for data transmission with RL. The EBR-RL protocol has improved performance in terms of the lifetime of the network and energy consumption when compared with baseline energy-efficient routing protocols.

Obi et al. [9] designed a lifetime aware centralized Q-routing protocol (LACQRP) for WSNs. The sink generates all possible MSTs of the network, which are used as the RTs. Q-learning is deployed on the sink to learn the optimal RT that maximizes the network lifetime. Simulation results show that the LACQRP converges to the optimal RT and has a better network lifetime when compared with RLBR and R2LTO. However, the LACQRP's computational complexity increases exponentially with the number of network sensor nodes. This makes the LACQRP inapplicable for large WSNs.

Obi et al. [13] proposed a Centralized Routing Protocol for Lifetime Optimization using a Genetic Algorithm and Reinforcement Learning (CRPLOGARL) that solves the problem of NP-hardness in the LACQRP. The CRPLOGARL also uses Q-learning to learn the MSTs that maximize the network lifetime. Simulation results show that the CRPLOGARL can provide a suboptimal routing path when compared to the LACQRP with less computational time. However, due to the large state space and action space of the LACQRP and CRPLOGARL, the baseline Q-learning used to implement these protocols suffers from degradation in the convergence speed and network lifetime due to the large number of learning episodes required to learn the optimal routing path.

The comparison of the related works is shown in Table 1.

Table 1. Comparison of the related works.

Routing Protocol	Objective	RL Technique	Control Technique	Drawback
Q-Routing [19]	Learns the optimal paths to minimizes the packet delivery delay.	Q-learning	Distributed	i. Requires Q-value freshness. ii. Sensitivity to parameter setting. iii. Slow convergence to optimal routing paths.
RL-based constrained flooding [20]	Optimizes the cost of constrained flooding (delivery delay, hop count).	Q-learning	Distributed	Degradation in packet delivery delay when compared with direct routing.
AdaR [24]	Maximizes network lifetime taking into consideration the hop count, node residual energy, link reliability, and the number of paths crossing a node.	LSPI	Distributed	i. No explicit definition of the network lifetime. ii. High computation complexity.
Energy-aware selfishness RL-based routing [25]	Minimizes the energy consumption.	Q-learning	Distributed	The selfishness and energy functions were not provided.
RLGR [26]	Improved the network lifetime by learning the optimal routing paths with factors such as hop count and node residual energy.	Q-learning	Distributed	Slow convergence to the optimal routing paths.
Q-PR [28]	Maintains the trade-off between network lifetime and the expected the number of retransmissions while increasing the packet delivery ratio.	Q-learning	Distributed	i. The message's importance is not balanced with the energy cost of using a constant a discount factor of one. ii. The selection of the next forwarder requires the requisites of neighbors. iii. Non-refinement of the estimation of the residual energy of the sensor nodes.
RL-based balancing energy routing [29]	Balancing the trade-off of minimizing energy consumption and maximizing the network lifetime by selecting routing paths based on the energy consumption of paths and residual energy of nodes.	Q-learning	Distributed	The network lifetime is the time when the the first node depletes its energy source, however, sensing is still possible unless the node is the sink.
E-FROMS [30]	Balances the energy consumption in multiple sinks by learning the optimal spanning tree that minimizes the energy-based reward.	Q-learning	Distributed	The state space and action space overhead are high and very high respectively.
QELAR [31]	Increases the network lifetime by finding the optimal routing path from each sensor node to the sink and distribute the residual the energy of each sensor node evenly.	Q-learning	Distributed	i. High overhead due to control packets. ii. Slow convergence to the optimal routing paths.

Table 1. *Cont.*

Routing Protocol	Objective	RL Technique	Control Technique	Drawback
RL-based routing interacting with WSN with moving vehicles [32]	Learn the routing paths between sensor nodes and moving sinks taking into consideration of hop count and energy signal strength to maximize the network lifetime.	Q-learning	Distributed	High overhead due to control packets.
OPT-EQ-Routing [33]	Optimizes the network lifetime while minimizing the control overhead by balancing the routing load among the sensor nodes taking into consideration the sensor nodes' current residual energies.	Q-learning	Distributed	Requires too many iterations to converge to the optimal paths.
EQR-RL [34]	Minimizing the network energy consumption while ensuring the packet delivery delay by learning the optimal routing path taking into consideration the residual energy of the next forwarder, the ratio of packets between the packet sender to the selected forwarder, and link delay.	Q-learning	Distributed	High convergence time to the optimal route.
RLLO [35]	Maximizing the network's lifetime and packet delay by learning the routing paths using the node residual energy and hop counts to the sink in the reward function.	Q-learning	Distributed	Very high probability of network isolation.
QSGrd [38]	Minimizing the energy consumption of the sensor nodes by jointly using Q-learning and transmission gradient.	Q-learning	Distributed	i. Slow convergence to the optimal routing paths. ii. The static parameter of the Q-learning leads to network performance degradation. iii. Increased computation time.
MRL-SCSO [39]	Maximizes the network lifetime by learning the next forwarder taking into account buffer lengthand node residual energy. Incorporating a sleeping schedule decreases the energy consumption of nodes.	Q-learning	Distributed	Increased number of episodes to learn the network.

Table 1. *Cont.*

Routing Protocol	Objective	RL Technique	Control Technique	Drawback
RLBR [41]	Search for optimal paths taking into consideration of hop count, link distance, and residual energy.	Q-learning	Distributed	Slow convergence to the optimal routing paths.
R2LTO [42]	Learns the optimal paths to the sink by considering the hop count, residual energy, and transmission energy between nodes.	Q-learning	Distributed	Slow convergence to the optimal routing paths.
RL-based routing protocol [43]	Chooses the next forwarder with Q-learning by using the the inverse of the distance between connected sensor nodes.	Q-learning	Distributed	Increased number of episodes to learn the network.
EBR-RL [45]	Learns the optimal routing path using hop count and the residual energy of sensor nodes to maximize the network lifetime.	Q-learning	Distributed	Slow convergence to the optimal routing paths.
LACQRP [9]	Learn the optimal MST that maximizes the network lifetime.	Q-learning	Centralized	Computational complexity increases exponentially with the number of sensor nodes.
CRPLOGARL [13]	Learn the optimal or near-optimal MST that maximizes the network's lifetime.	Q-learning	Centralized	Slow convergence to the optimal or near-optimal MST.

3. Methodology

A WSN consists of a set of sensor nodes and a sink. When the network has been set up and initialized, all the sensor nodes broadcast their status information to the sink. The status information of each sensor node broadcasted to the sink includes a distinctive identification (ID), $x - y$ cartesian coordinates, traffic load (number of data packets to send in each second to the sink), initial residual energy, and maximum communication range. The graph of the WSN $G = (V, E)$, is built after the sink has received all the sensor nodes' status information. $V = \{v_1, \ldots, v_n\}$ is the sensor nodes set and $E = \{e_1, \ldots, e_m\} \subseteq V \times V$ is the WSN links set. Each link is the connection between two sensor nodes in the WSN. Two sensor nodes are only connected if their cartesian distance is equal to or less than the sensor nodes' maximum communication range. The list of routing tables (RTs) is computed by the sink with GA-based MSTs when the edge weight is the cartesian distance between two connected sensor nodes. The sink selects an RT in each learning round of data transmission with an LSPI and broadcasts it to all sensor nodes for data transmission. This enables the global energy consumption and network lifetime optimization of the WSN. The centralized routing protocol for energy and lifetime optimization using GA and LSPI is presented in the sequel.

3.1. A GA-Based MSTs

The GA-based MSTs is extracted from the CRPLOGARL [13] for WSNs and is presented in the sequel.

Consider an integer weight $\mathsf{w}(\mathsf{e}) > 0$ that is associated with an edge $\mathsf{e} \in \mathsf{E}$. Let $\mathsf{w}(\mathsf{T})$ be the sum of the edge weights in an MST of the network graph. The computational complexity of finding all MSTs of the network graph is exponential (that is, the problem is NP-hard). The approach GA is used to alleviate the NP-hardness of the All-MST algorithm. The distance-based MST is considered and is defined as the spanning tree having the minimum possible total cartesian distances between the connected vertices.

The GA population is obtained from MSTs calculated with a baseline MST algorithm [46]. The baseline algorithms for finding the MST of a connected undirected graph are Prim's algorithm [47], Kruskal's algorithm [48], and Boruka's algorithm [46]. Boruka's algorithm and Kruskal's algorithm normally yield a network graph with only one MST. The reason is that Boruka's and Kruskal's algorithms add the least edge to an existing tree while looking at the whole network graph until the MST is obtained. Subsequently, Prim's algorithm finds an MST by inputting any node as the root. This enables Prim's algorithm to generate more than one MST for a network graph that does not have distinct edge weights when using different nodes as the root [49]. Therefore the maximum MSTs that can be generated by Prim's algorithm for a network graph when varying the root node is the number of the graph nodes n.

The GA-based MSTs extracts different MSTs of the network graph as the initial population with the baseline Prim's algorithm by varying the root node. Prim's algorithm for generating the initial population runs in $O(\mathsf{N}\mathsf{m}\log\mathsf{n})$ time, where n, m, and N is the number of nodes, edges, and MSTs of the network graph G, respectively. n is the upper bound of N. The algorithm for finding the GA initial population is given in Algorithm 1.

Algorithm 1 Algorithm to generate initial population for GA-based MSTs.

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

The GA for generating MSTs for the number of generations, NG specified, is given in Algorithm 2.

A chromosome in the initial population is an MST, with its genes as the graph edges [50]. The mutation operator and the crossover operator are used to evolve the initial population. This results in the generation of new MSTs as a result of the inheritance of some parents' edges.

The fitness of the newly formed individual, T^k is measured with the objective function, which is the cost of the MST, T^* of the network graph as given in Equation (20).

$$fitness(k) = \text{Poss}\left[\sum_{u,v\in T^k} d_{u,v} = \sum_{u,v\in T^*} d_{u,v}\right] \tag{20}$$

where $d_{u,v}$ is given in Equation (21).

$$d_{u,v} = \sqrt{(x(u) - x(v))^2 + (y(u) - y(v))^2} \tag{21}$$

Algorithm 2 GA for generating MSTs.

Input: mr, cr, NG
Output: MSTs

$\mathcal{P} = \{\}$
Form $k \leq n$ distinctive MSTs with **Algorithm 2**
$\mathcal{P} \leftarrow k$ unique MSTs
for i $= 1$ to NG **do**
  $\mathcal{P}_i = \{\}$
  $\mathsf{n_c} = \frac{100}{\mathsf{cr}}$
  for j $= 1$ to $\mathsf{n_c}$ **do**
    T1, T2 $\in_R \mathcal{P}$
    G1 $=$ T1 $\cup$ T2
    T $= Prim($G1, v$)$
    if T $\notin \mathcal{P}_i$ **then**
      $\mathcal{P}_i \leftarrow$ T
    end if
  end for
  $\mathsf{n_m} = \frac{100}{\mathsf{mr}}$
  for j $= 1$ to $\mathsf{n_m}$ **do**
    T3 $\in_R \mathcal{P}$
    $\mathsf{e_{i,j}} \in_R$ T3
    G2 $=$ T3 $- \mathsf{e_{i,j}}$
    $\mathsf{G^*} =$ G $- \mathsf{e_{i,j}}$
    Cut Set $= \{\mathsf{e_{i,j}} \mid \mathsf{e_{i,j}} \in \mathsf{G^*}$ & $\mathsf{e_{i,j}} \notin$ G2$\}$
    $\mathsf{e^*_{i,j}} \in_R$ Cut Set
    $\mathsf{G2^*} =$ G2 $+ \mathsf{e^*_{i,j}}$
    if $\mathsf{G2^*}$ is a tree of G **then**
      if $\mathsf{G2^*} \notin \mathcal{P}_i$ **then**
        $\mathcal{P}_i \leftarrow \mathsf{G2^*}$
      end if
    end if
  end for
  for T in $\mathcal{P}_i$ **do**
    Calculate $fitness$ with Equation (20)
    if $fitness$ is True **then**
      if T $\notin \mathcal{P}$ **then**
        $\mathcal{P} \leftarrow$ T
      end if
    end if
  end for
end for
Return $\mathcal{P}$

The mutation process is achieved by randomly selecting an edge $\mathsf{e_{i,j}}$ from a chosen MST T3 contained in the population and removing $\mathsf{e_{i,j}}$ from T3 and the original network graph G to create the sub-graphs G2 and $\mathsf{G^*}$, respectively. A random edge $\mathsf{e^*_{i,j}}$ contained in the cut set of $\mathsf{G^*}$ is added to G2 to form a new sub-graph, $\mathsf{G2^*}$ [51].

The cut set of $\mathsf{G^*}$ is the set of edges contained in $\mathsf{G^*}$ and are not contained in G2. The individual formed by the mutation process has to be a tree of G before it is accepted, as illustrated in Figure 2. The number of times to apply the mutation operator before choosing the fitness individual is specified by the mutation rate *mr*.

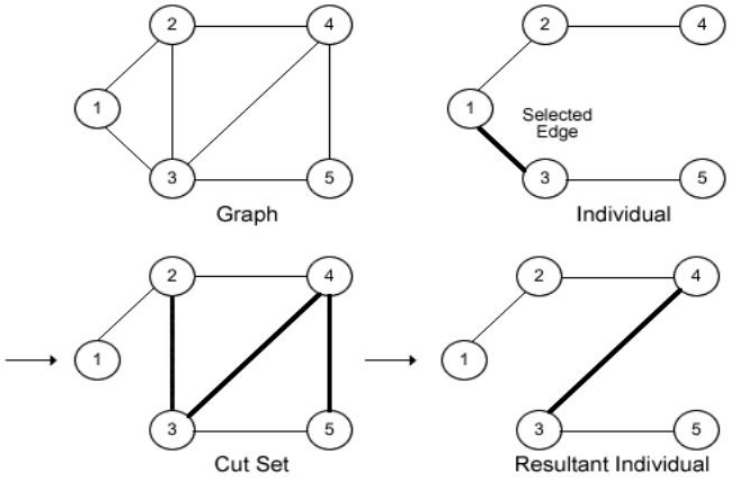

Figure 2. MST Mutation.

The crossover process is achieved by selecting two individuals T1 and T2 randomly from the population as parents to create offspring for the subsequent generation. T1 and T2 are joined to create a sub-graph G1 of G by passing the union operation [51]. The individual form is an MST of G1 and the network graph G as shown in Figure 3. The number of times the crossover operator is carried out is determined by the crossover rate *cr*.

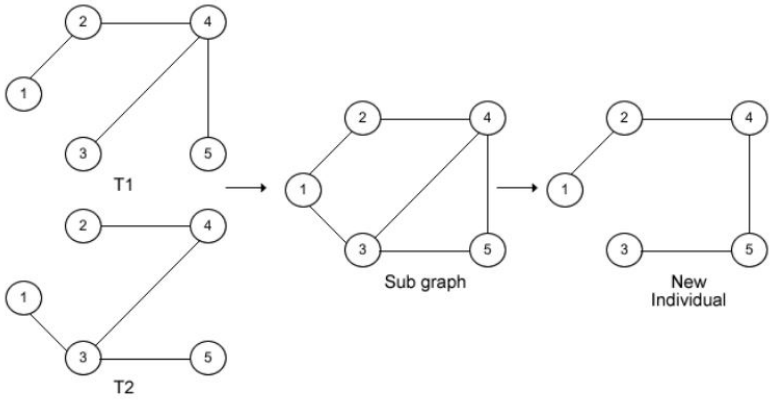

Figure 3. Crossover between two MSTs T1 and T2.

3.2. A Centralized Routing Protocol for Lifetime and Energy Optimization Using GA and LSPI

After the network initialization, the sink generates possible distance-based MSTs using the GA-based MSTs. The generated MSTs are used as the routing tables (RTs) by the CRPLEOGALSPI. The optimization problem addressed by the 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.

The optimization problem is to find the RT such that:

$$\max_{v \in V}\left\{ESRE_v\right\} + \min_{v,u \in V}\left\{\sum_{u=0}^{|V|}\sum_{v=0}^{|V|} e_{u,v}EC_{u,v}\right\} subjectto \tag{22a}$$

$$\sum_{u=0,u\neq v}^{|V|} e_{u,v} = 1 \tag{22b}$$

$$\sum_{v=0,u\neq v}^{|V|} e_{u,v} = 1 \tag{22c}$$

$$d_{u,v} \leq R_{max} \tag{22d}$$

$$e_{u,v} \in \{0,1\} \tag{22e}$$

Equation (22a) is the objective function and Equations (22b)–(22e) 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 (22a) is used to maximize the network lifetime, while the second term is used to minimize the network energy consumption. The constraints in Equation (22b) and (22c) ensure that the routing path is a tree. The constraint in Equation (22d) enables transmission between two connected sensor nodes whose link distance is less than or equal to the maximum network transmission radius.

Therefore, the CRPLEOGALSPI tries to find the RT that maximizes the network lifetime while minimizing the network energy consumption using LSPI. The network lifetime considered in the design of 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 the sink for data transmission. The CRPLEOGALSPI is deployed at the sink.

The solving process of the objective function subject to the constraints is given in the design of the proposed protocol using LSPI and is provided in the sequel.

The state space and action space of the agent are the MSTs generated by the GA-based MSTs. For a current MST, s, being used by the sink in receiving data from the sensor nodes, the action of the agent 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 of the energy 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 set 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 (23)

$$\hat{Q}(s,a) = w_1 EC_{max}(s,a) + w_2 EC_{sum}(s,a) = w\varphi(s,a)^T \tag{23}$$

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 (23) is approximated using a set of samples $\mathcal{D} = \left\{(s_i, a_i, r_i, \hat{s}_i)|i = 1, 2 \ldots, M\right\}$ as given in Equation (24).

$$\hat{w} = \hat{X}^{-1}\hat{y} \tag{24}$$

The set of samples $\mathcal{D}$ is obtained by taking a random action in a given state.
The algorithm used to generate the random samples by the sink is given in Algorithm 3.

Algorithm 3 Samples Generation Algorithm.

Input: List of RTs, Maximum number of samples, $|\mathcal{D}|_{max}$, Number of iterations, N
Output: Set of samples, $\mathcal{D}$
1: $\mathcal{D} = \{\}$
2: Initialize a random RT as s_0
3: **for** $i = 1$ to N **do**
4: $s_i = s_o$
5: The agent Choose a random RT, a_i
6: The agent receives a reward, r_i using Equation (27)
7: $s_i \leftarrow s_o$
8: $\hat{s}_i = s_o$
9: **if** $(s_i, a_i, \hat{s}_i, r_i) \notin \mathcal{D}$ **then**
10: $\mathcal{D} \leftarrow (s_i, a_i, \hat{s}_i, r_i)$
11: **end if**
12: **if** $|\mathcal{D}| = |\mathcal{D}|_{max}$ **then**
13: **break**
14: **end if**
15: **end for**
16: Return $\mathcal{D}$

Assuming $\hat{X}^0 = 0$ and $\hat{y}^0 = 0$ initially, Equations (25) and (26) will give the approximated values of $\hat{X}^{i+1}$ and $\hat{y}^{i+1}$, respectively, of a new sample $(s_i, a_i, r_i, \hat{s}_i)$

$$\hat{X}^{i+1} \leftarrow \hat{X}^i + \varphi(s_i, a_i)\left[\varphi(s_i, a_i) - \gamma\varphi(\hat{s}_i, \underset{a \in A}{\operatorname{argmin}}(\hat{s}_i, a))\right]^T \tag{25}$$

where γ is the discount factor.

$$\hat{y}^{i+1} \leftarrow \hat{y}^i + \varphi(s_i, a_i)r_i \tag{26}$$

The reward obtained is modeled as in Equation (27)

$$r_i = \max_{v \in V}\{EC_v\} + \operatorname{sum}_{v \in V}\{EC_v\} \tag{27}$$

where EC_v is the energy consumption by the vth sensor node.

The sink evaluates the EC_v after each learning round using the difference between the formerly estimated sensor residual energy $ESRE_v^{Previous}$ and the later estimated sensor residual energy, $ESRE_v^{Current}$. Therefore EC_v is as given in Equation (28).

$$EC_v = ESRE_v^{Former} - ESRE_v^{Later} \tag{28}$$

The network lifetime is maximized while minimizing the energy consumption of the WSN by learning the RT that minimizes the Q-value using the epsilon-greedy technique. Given a number, $z \in (0, 1)$ randomly generated in each learning round and a probability

epsilon value, $\epsilon \in [0, 1]$, the agent selects its action in each learning round with the policy given in Equation (29).

$$a_t = \begin{cases} \underset{a}{\text{argmin}}(\varphi(s,a)^T w), \; if \; z < 1 - \epsilon \\ Random \; action, \; otherwise. \end{cases} \tag{29}$$

The designed CRPLEOGALSPI for learning the optimal RT to maximize network lifetime and minimize the network energy consumption is given in Algorithm 4. The novelty of the proposed protocol is in the use of the LSPI, which tends to converge faster to the best RT in solving the objective function subject to constraints when compared to Q-learning.

Algorithm 4 CRPLEOGALSPI.

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

Sink executes **Algorithm 2** to generate List of RTs
Sink collect set of samples, $\mathcal{D}$ with **Algorithm 3**
Initialize s_0 as a random RT
Initialize $\hat{X}^0 \leftarrow 0$ and $\hat{y}^0 \leftarrow 0$
Initialize weight, $w_o \leftarrow 0$
$\hat{w} \leftarrow w_o$
repeat
  $w \leftarrow \hat{w}$
  for $(s_i, a_i, r_i, \hat{s_i}) \in \mathcal{D}$ **do**
    $\hat{X}^{i+1} \leftarrow \hat{X}^i + \varphi(s_i, a_i)\left[\varphi(s_i, a_i) - \gamma\varphi(\hat{s_i}, \underset{a \in A}{\text{argmin}}(\hat{s_i}, a))\right]^T$
    $\hat{y}^{i+1} \leftarrow \hat{y}^i + \varphi(s_i, a_i) r_i$
  end for
  $\hat{w} = \hat{X}^{-1}\hat{y}$
until $||w - \hat{w}|| < \varepsilon$
return w
for $t = 1$ to L **do**
  $s_t = s_o$
  Sink chooses an RT using Equation (29)
  Sink receives data from the sensors using the chosen RT.
  Updates s_o as the current RT.
  if Any sensor node dies, **then**
    Delete the dead sensor node(s) from $G(V, E)$
    Delete edges connected to the dead sensor node(s)
    Rebuild $\mathsf{G}(\mathsf{V}, \mathsf{E})$
    if $\mathsf{G}(\mathsf{V}, \mathsf{E})$ is connected, **then**
      Do steps 1 to 20
    else
      break
    end if
  end if
end for

3.3. Energy Consumption Model

The energy consumed by the vth sensor node, EC_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 (30).

$$EC_v(p, d) = E_{tx}(p, d) + E_{rx}(p) \tag{30}$$

The energy consumed by a sensor node for sending and receiving data packets is given in Equations (31) and (33), respectively [33].

$$E_{tx}(p,d) = \begin{cases} E_{elec}p\ell + E_{fs}pd^2 \ \ if\ d \leqslant d_o \\ elecp\ell + E_{mp}pd^4 \ \ if\ d > d_o \end{cases} \tag{31}$$

$$E_{rx}(p) = E_{elec}p\tau \tag{32}$$

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 transmission energy, $E_{rx}(p)$ is the reception energy, and 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 $d \leqslant d_o$, 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:

$$d_o = \sqrt{\frac{E_{fs}}{E_{mp}}} \tag{33}$$

4. Simulation and Results Discussion

The performance of the GA-based MSTs is first established for convergence using simulation. Secondly, the performance analysis of the proposed routing protocol is achieved with simulations using the performance metrics of network lifetime, number of alive sensor nodes (NAN), energy consumption, and computation time. These performance metrics of the CRPLEOGALSPI are compared with that of the recent CRPLOGARL [13] as a way of validation. The lifetime of the network is calculated as the time taken for the sink not to be reachable by the alive sensor node(s). The NAN is the number of alive sensor nodes at the lifetime of the network. The CRPLEOGALSPI and CRPLOGARL are coded with python 3.8 under the "PyCharm" development environment. The graphical layout of the WSN is implemented with the python NetworkX module [52]. The python codes are 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 carry out the performance analysis and the network graph of the deployed WSN are shown in Table 2 and Figure 4, respectively.

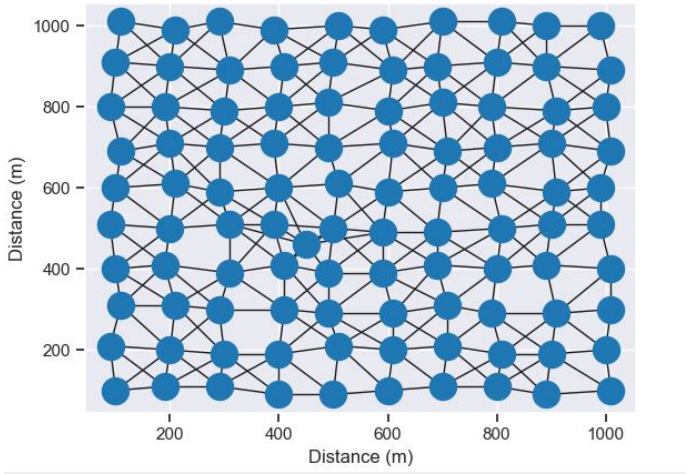

Figure 4. Network Graph of the deployed WSN.

Table 2. Simulation Parameters.

Parameters	Values
Number of sink	1
Number of sensors	100
Deployment Area of WSN	1000 m × 1000 m
Deployment of Sensor nodes	Random
$x - y$ coordinate of sink	$(500, 500)$
Maximum transmission range	150 m
Bandwidth of links	1 kbps
Size of data packet	1024 bits
Sensors initial residual energy	1 J to 10 J
Rate of packet generation	1/s to 10/s
e_{mp}	0.0013 pJ/bit/m^4
e_{fs}	10 pJ/bit/m^2
E_{elec}	50 nJ/bit
Discount factor	0.9
Epsilon	0.1
Sample size	100
Maximum generations	1000
Rate of crossover	0.1
Rate of Mutation	1

The performance of the GA-based MSTs for the deployed network graph of the WSN given in Figure 4 is shown in Figure 5.

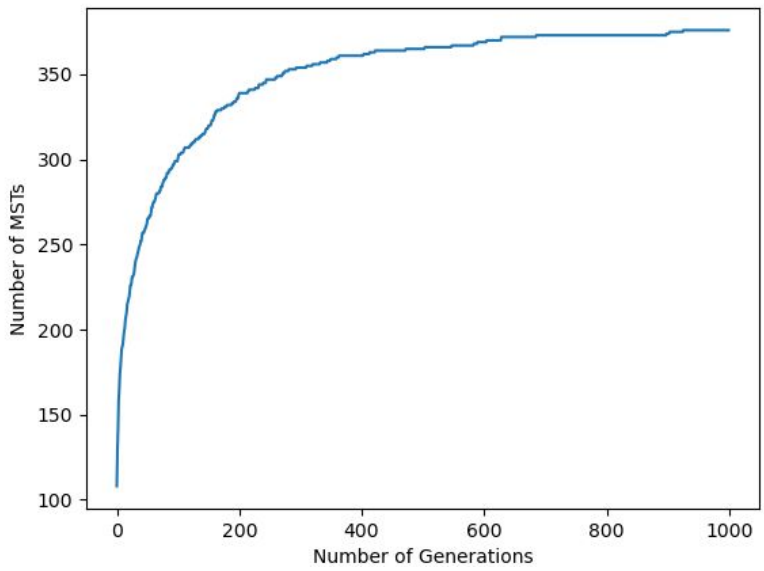

Figure 5. Number of MSTs with Number of Generations.

As seen in Figure 5, the GA-based MSTs is able to find 376 MSTs of the network graph for 1000 generations. This also implies the convergence of the solution of the GA-based MSTs in a reasonable number of generations. The GA-based MSTs can work for all MSTs problems in polynomial time.

The number of alive sensor nodes of the WSN at each round of data transmission of the proposed routing protocol, CRPLEOGALSPI, as compared with CRPLOGARL 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. As seen in Figure 6, 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. When the network graph is disconnected, there is no path to reach the sink located at the $x - y$ coordinates of $(500, 500)$, as shown in Figure 4. 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. 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, which maximizes the network lifetime while minimizing the network energy consumption when compared to CRPLOGARL. This shows that the LSPI used in implementing CRPLEOGAL-SPI utilizes data effectively and efficiently when compared to the baseline Q-learning used to implement the CRPLOGARL.

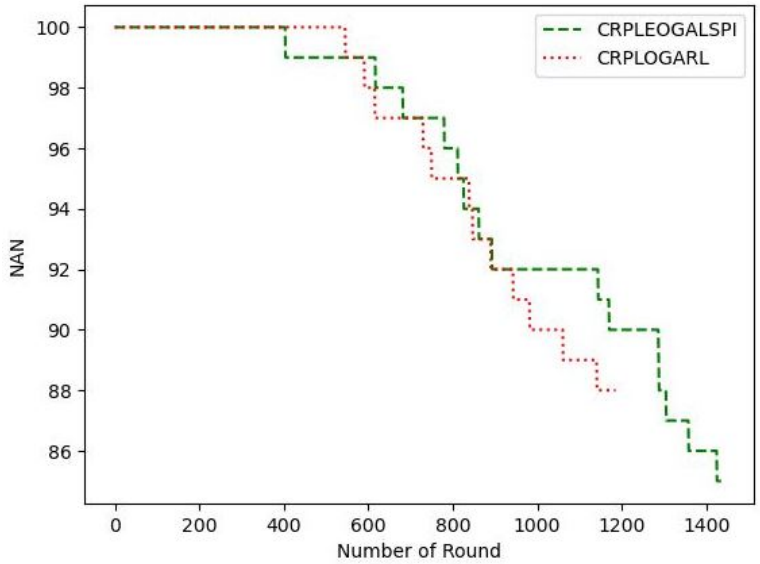

Figure 6. Number of alive sensors with Round of Data Transmission.

The time for the sink not to be reachable by the alive sensor nodes of the CRPLEOGAL-SPI is compared with the CRPLOGARL for increasing sensor nodes' initial energy as shown in Figure 7. 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.

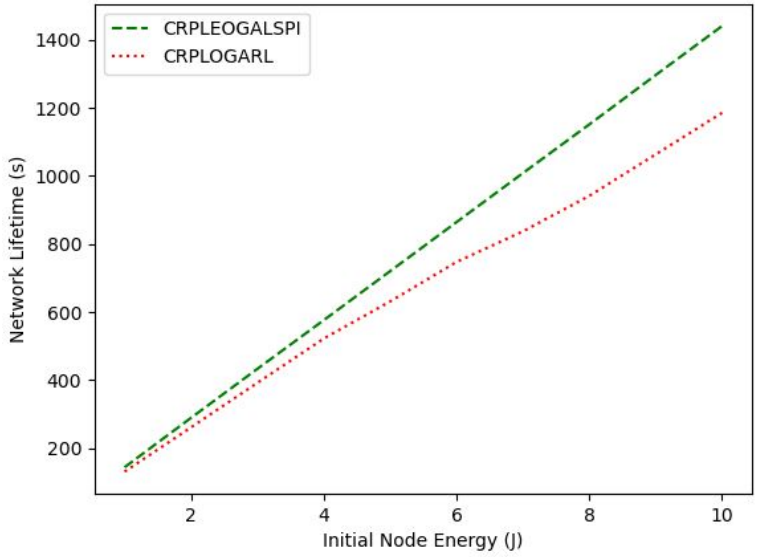

Figure 7. Network lifetime with Initial Node Energy.

The proposed CRPLEOGALSPI has an improved network lifetime performance of 18.04% when compared to the CRPLOGARL with increasing sensor node residual energy. This is because the CRPLEOGALSPI utilizes LSPI, which is more data efficient and converges faster to the RT, which balances the energy consumption among the sensor nodes and minimizes the network energy consumption as shown in Figure 8. This is against the CRPLOGARL 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 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. Subsequently, the CRPLEOGALSPI has a reduced energy consumption of 58.96% when compared to CRPLOGARL for increasing the initial energy of the sensor nodes. This is because CRPLEOGALSPI optimizes the network lifetime and the network energy consumption using LSPI. This is against the CRPLOGARL, which optimizes only the network lifetime using Q-learning. This also implies that CRPLEOGALSPI has a reduced CO_2 footprint when compared to CRPLOGARL 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 CRPLOGARL for an increased initial sensor node energy, as shown in Figure 9. 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).

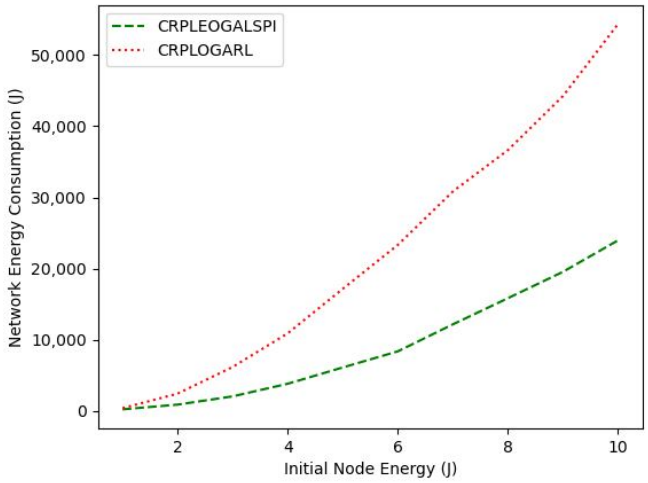

Figure 8. Network Energy Consumption with Initial Node Energy.

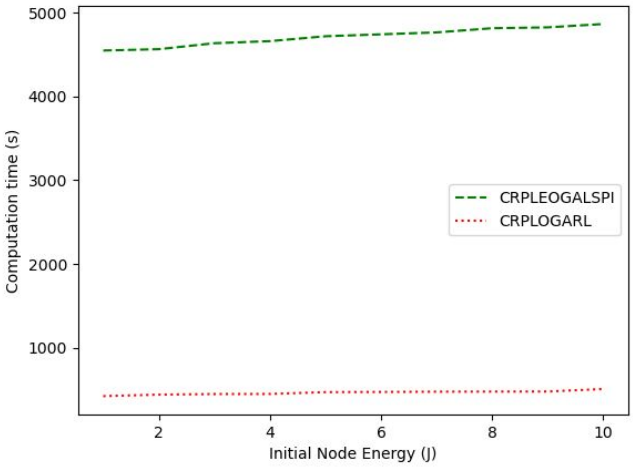

Figure 9. Computation time with Initial Node Energy.

The time for the sink not to be reachable by alive sensor nodes of the CRPLEOGALSPI is compared with the CRPLOGARL for increasing sensor nodes' packet generation rate as shown in Figure 10. 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 CRPLOGARL with an increasing packet generation rate of the sensor nodes.

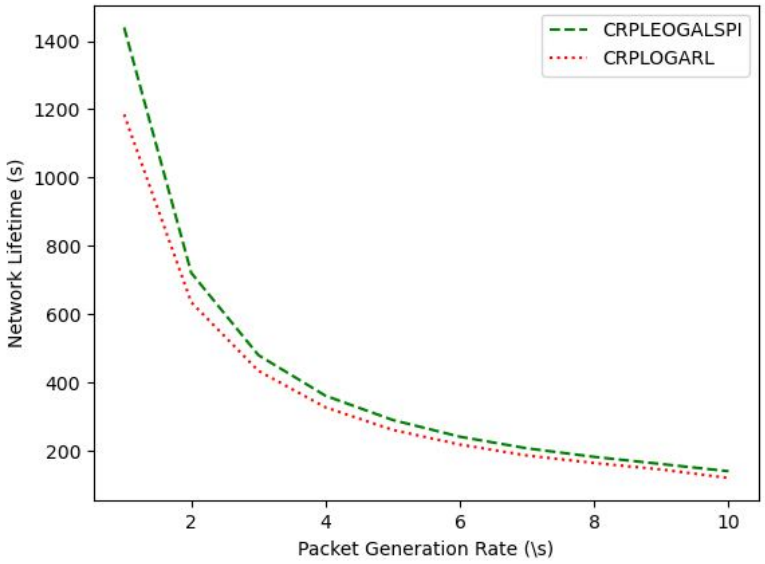

Figure 10. Network lifetime with Packet Generation Rate.

This is because CRPLEOGALSPI utilizes LSPI, which is more data efficient and converges faster to the RT, which balances the energy consumption among the sensor nodes and minimizes the network energy consumption, as shown in Figure 11. This is against the CRPLOGARL 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 energy. Subsequently, the CRPLEOGALSPI has a reduced energy consumption of 56.36% when compared to the CRPLOGARL for increasing the packet generation rate of the sensor nodes. This is because the CRPLEOGALSPI optimizes the network lifetime and the network energy consumption using LSPI. This is contrasted with the CRPLOGARL, 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 CRPLOGARL for an increased sensor node packet generation rate. This is because that CO_2 footprint is directly proportional to the network energy consumption.

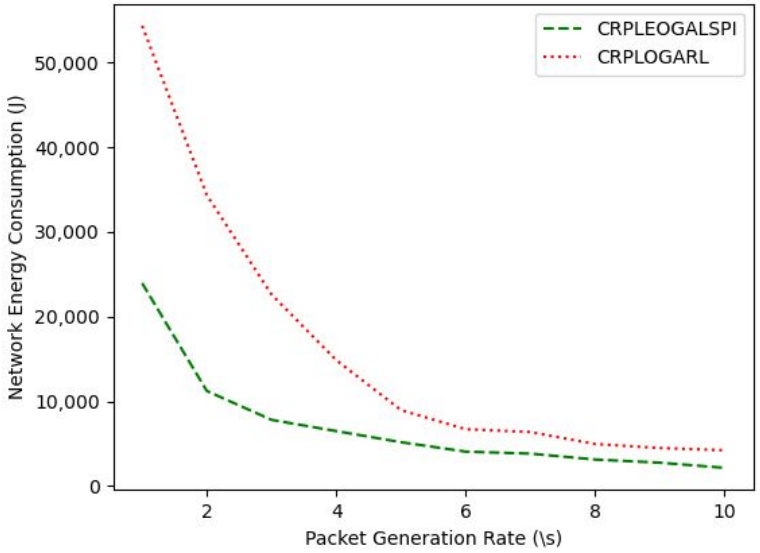

Figure 11. Network Energy Consumption with Packet Generation Rate.

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 CRPLOGARL for an increased packet generation rate of the sensor node, as shown in Figure 12. 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).

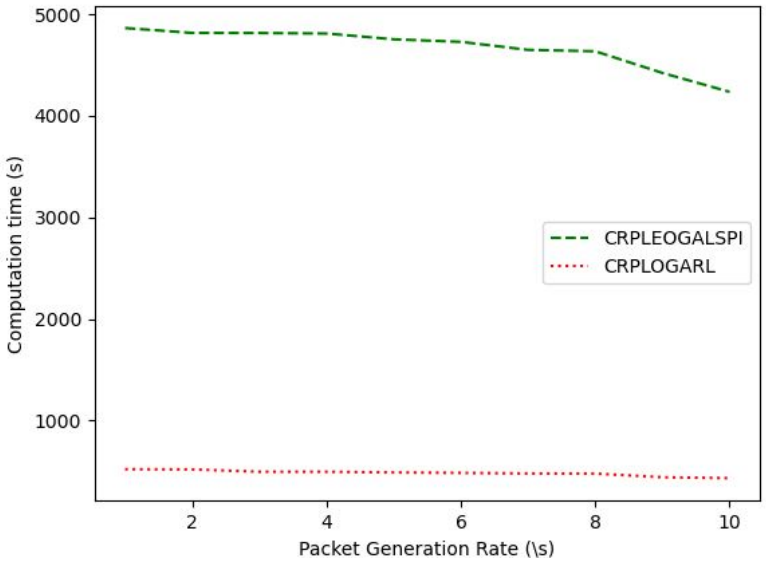

Figure 12. Computation time with Packet Generation Rate.

5. Conclusions

This paper presented the design of a centralized routing protocol for 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 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 lifetime and energy optimization using a GA and LSPI improves network lifetime and energy consumption

when compared with the existing 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 proposed protocol comes with an increased computation time. Future work will consider emulating the proposed protocol using Mininet, considering the real-world parameters of a typical WSN, and taking into consideration the cost of control packets and prioritization schemes when some sensor nodes send more data packets than others.

Author Contributions: Conceptualization, methodology, validation, writing—original draft preparation, E.O.; software, resources, E.O. and O.E.O.; writing—review and editing, O.E.O.; supervision, project administration, Z.M. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Nigerian Petroleum Technology Trust Fund (PTDF) Overseas Scholarship Scheme and by Paul Sabatier University.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: The Observatory of Indexing and Multimedia Information Research Systems (OSIRIM) platform used for the research experiments was provided by the Toulouse Institute for Research in Computer Science (IRIT).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Priyadarshi, R.; Gupta, B.; Anurag, A. Deployment techniques in wireless sensor networks: A survey, classification, challenges, and future research issues. *J. Supercomput.* **2020**, *76*, 7333–7373. [CrossRef]
2. Rawat, P.; Singh, K.D.; Chaouchi, H.; Bonnin, J.M. Wireless sensor networks: A survey on recent developments and potential synergies. *J. Supercomput.* **2014**, *68*, 1–48. [CrossRef]
3. Matin, M.A.; Islam, M.M. Overview of wireless sensor network. *Wirel. Sens. Netw.-Technol. Protoc.* **2012**, *1*, 1–24.
4. Xia, F. Wireless sensor technologies and applications. *Sensors* **2009**, *9*, 8824–8830. [CrossRef] [PubMed]
5. Engmann, F.; Katsriku, F.A.; Abdulai, J.D.; Adu-Manu, K.S.; Banaseka, F.K. Prolonging the lifetime of wireless sensor networks: A review of current techniques. *Wirel. Commun. Mob. Comput.* **2018**, 1–23 [CrossRef]
6. Nayak, P.; Swetha, G.K.; Gupta, S.; Madhavi, K. Routing in wireless sensor networks using machine learning techniques: Challenges and opportunities. *Measurement* **2021**, *178*, 1–15 [CrossRef]
7. Al Aghbari, Z.; Khedr, A.M.; Osamy, W.; Arif, I.; Agrawal, D.P. Routing in wireless sensor networks using optimization techniques: A survey. *Wirel. Pers. Commun.* **2020**, *111*, 2407–2434. [CrossRef]
8. Mostafaei, H.; Menth, M. Software-defined wireless sensor networks: A survey. *J. Netw. Comput. Appl.* **2018**, *119*, 42–56. [CrossRef]
9. Obi, E.; Mammeri, Z.; Ochia, O.E. A Lifetime-Aware Centralized Routing Protocol for Wireless Sensor Networks using Reinforcement Learning. In Proceedings of the 17th International Conference on Wireless and Mobile Computing, Networking and Communications, Bologna, Italy, 11–13 October 2021; pp. 363–368.
10. Sutton, R.S.; Barto, A.G. *Reinforcement Learning: An Introduction*, 2nd ed.; MIT press: Cambridge, MA, USA; London, UK, 2018; pp. 119–138.
11. Yamada, T.; Kataoka, S.; Watanabe, K. Listing all the minimum spanning trees in an undirected graph. *Int. J. Comput. Math.* **2010**, *87*, 3175–3185. [CrossRef]
12. Whitley, D. A genetic algorithm tutorial. *Stat. Comput.* **1994**, *4*, 65–85. [CrossRef]
13. Obi, E.; Mammeri, Z.; Ochia, O.E. 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, 16–20 October 2022; pp. 5–12.
14. Watkins, C.J.; Dayan, P. Q-learning. *Mach. Learn.* **1992**, *8*, 279–292. [CrossRef]
15. Lagoudakis, M.G.; Parr, R. Least-squares policy iteration. *J. Mach. Learn. Res.* **2003**, *4*, 1107–1149.
16. Kaelbling, L.P.; Littman, M.L.; Moore, A.W. Reinforcement learning: A survey. *J. Artif. Intell. Res.* **1996**, *4*, 237–285. [CrossRef]
17. Mammeri, Z. Reinforcement learning based routing in networks: Review and classification of approaches. *IEEE Access* **2019**, *7*, 55916–55950. [CrossRef]
18. Bradtke, S.J.; Barto, A.G. Linear least-squares algorithms for temporal difference learning. *Mach. Learn.* **1996** *22*, 33–57. [CrossRef]

19. Boyan, J.; Littman, M. Packet routing in dynamically changing networks: A reinforcement learning approach. *Adv. Neural Inf. Process. Syst.* **1993**, *6*, 671–678.
20. Zhang, Y.; Fromherz, M. Constrained flooding: A robust and efficient routing framework for wireless sensor networks. In Proceedings of the 20th International Conference on Advanced Information Networking and Applications-Volume 1, Vienna, Austria, 18–20 April 2006; pp. 1–6.
21. Maroti, M. Directed flood-routing framework for wireless sensor networks. In Proceedings of the ACM/IFIP/USENIX International Conference on Distributed Systems Platforms and Open Distributed Processing, Berlin, Germany, 18–20 October 2004; pp. 99–114.
22. He, T.; Krishnamurthy, S.; Stankovic, J.A.; Abdelzaher, T.; Luo, L.; Stoleru, R.; Yan, T.; Gu, L.; Hui, J.; Krogh, B. Energy-efficient surveillance system using wireless sensor networks. In Proceedings of the 2nd International Conference on Mobile Systems, Applications, and Services, Boston, MA, USA, 6–9 June 2004; pp. 270–283.
23. Intanagonwiwat, C.; Govindan, R.; Estrin, D. Directed diffusion: A scalable and robust communication paradigm for sensor networks. In Proceedings of the 6th Annual International Conference on Mobile Computing and Networking, Boston, MA, USA, 6–11 August 2000; pp. 56–67.
24. Wang, P.; Wang, T. Adaptive routing for sensor networks using reinforcement learning. In Proceedings of the 6th IEEE International Conference on Computer and Information Technology, Seoul, Republic of Korea, 20–22 September 2006; p. 219.
25. Nurmi, P. Reinforcement learning for routing in ad hoc networks. In Proceedings of the 5th IEEE International Symposium on Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks and Workshops, Limassol, Cyprus, 16–20 April 2007; pp. 1–8.
26. Dong, S.; Agrawal, P.; Sivalingam, K. Reinforcement learning based geographic routing protocol for UWB wireless sensor network. In Proceedings of the IEEE Global Telecommunications Conference, Washington, DC, USA, 26–30 November 2007; pp. 652–656.
27. Karp, B.; Kung, H.T. GPSR: Greedy perimeter stateless routing for wireless networks. In Proceedings of the 6th Annual International Conference on Mobile Computing and Networking, Boston MA, USA, 6–11 August 2000; pp. 243–254.
28. Arroyo-Valles, R.; Alaiz-Rodriguez, R.; Guerrero-Curieses, A.; Cid-Sueiro, J. Q-probabilistic routing in wireless sensor networks. In Proceedings of the IEEE 3rd International Conference on Intelligent Sensors, Sensor Networks and Information, Melbourne, VIC, Australia, 3–6 December 2007; pp. 1–6.
29. Naruephiphat, W.; Usaha, W. Balancing tradeoffs for energy-efficient routing in MANETs based on reinforcement learning. In Proceedings of the VTC Spring IEEE Vehicular Technology Conference, Marina Bay, Singapore, 11–14 May 2008; pp. 2361–2365.
30. Förster, A.; Murphy, A.L. Balancing energy expenditure in WSNs through reinforcement learning: A study. In Proceedings of the 1st International Workshop on Energy in Wireless Sensor Networks, Santorini Island, Greece, 11–14 June 2008; pp. 1–7.
31. Hu, T.; Fei, Y. QELAR: A q-learning-based energy-efficient and lifetime-aware routing protocol for underwater sensor networks. In Proceedings of the IEEE International Performance, Computing and Communications Conference, Austin, TX, USA, 7–9 December 2008; pp. 247–255.
32. Yang, J.; Zhang, H.; Pan, C.; Sun, W. Learning-based routing approach for direct interactions between wireless sensor network and moving vehicles. In Proceedings of the 16th International IEEE Conference on Intelligent Transportation Systems, The Hague, The Netherlands, 6–9 October 2013; pp. 1971–1976.
33. Oddi, G.; Pietrabissa, A.; Liberati, F. Energy balancing in multi-hop Wireless Sensor Networks: An approach based on reinforcement learning. In Proceedings of the 2014 NASA/ESA IEEE Conference on Adaptive Hardware and Systems, Leicester, UK, 14–17 July 2014; pp. 262–269.
34. Jafarzadeh, S.Z.; Moghaddam, M.H.Y. Design of energy-aware QoS routing protocol in wireless sensor networks using reinforcement learning. In Proceedings of the 2014 IEEE 27th Canadian Conference on Electrical and Computer Engineering, Toronto, ON, Canada, 4–7 May 2014; pp. 1–5.
35. Guo, W.J.; Yan, C.R.; Gan, Y.L.; Lu, T. An intelligent routing algorithm in wireless sensor networks based on reinforcement learning. *Appl. Mech. Mater.* **2014**, *678*, 487–493. [CrossRef]
36. Shah, R.C.; Rabaey, J.M. Energy aware routing for low energy ad hoc sensor networks. In Proceedings of the IEEE Wireless Communications and Networking Conference Record, Orlando, FL, USA, 17–21 March 2002; pp. 350–355.
37. Yessad, S.; Tazarart, N.; Bakli, L.; Medjkoune-Bouallouche, L.; Aissani, D. Balanced energy-efficient routing protocol for WSN. In Proceedings of the IEEE International Conference on Communications and Information Technology, Hammamet, Tunisia, 26–28 June 2012; pp. 326–330.
38. Debowski, B.; Spachos, P.; Areibi, S. Q-learning enhanced gradient-based routing for balancing energy consumption in WSNs. In Proceedings of the IEEE 21st International Workshop on Computer Aided Modelling and Design of Communication Links and Networks, Toronto, ON, Canada, 23–25 October 2016; pp. 18–23.
39. Renold, A.P.; Chandrakala, S. MRL-SCSO: Multi-agent reinforcement learning-based self-configuration and self-optimization protocol for unattended wireless sensor networks. *Wirel. Pers. Commun.* **2017**, *96*, 5061–5079. [CrossRef]
40. Gnawali, O.; Fonseca, R.; Jamieson, K.; Moss, D.; Levis, P. Collection tree protocol. In Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems, Berkeley, CA, USA, 4–6 November 2009; pp. 1–14.
41. Guo, W.; Yan, C.; Lu, T. Optimizing the lifetime of wireless sensor networks via reinforcement-learning-based routing. *Int. J. Distrib. Sens. Netw.* **2019**, *15*, 1–20. [CrossRef]

42. Bouzid, S.E.; Serrestou, Y.; Raoof, K.; Omri, M.N. Efficient routing protocol for wireless sensor network based on reinforcement learning. In Proceedings of the 5th IEEE International Conference on Advanced Technologies for Signal and Image Processing, Sousse, Tunisia, 2–5 September 2020; pp. 1–5.
43. Sapkota, T.; Sharma, B. Analyzing the energy efficient path in Wireless Sensor Network using Machine Learning. *ADBU J. Eng. Technol.* **2021**, *10*, 1–7.
44. Intanagonwiwat, C.; Govindan, R.; Estrin, D.; Heidemann, J.; Silva, F. Directed diffusion for wireless sensor networking. *IEEE/ACM Trans. Netw.* **2003**, *11*, 2–16. [CrossRef]
45. Mutombo, V.K.; Shin, S.Y.; Hong, J. EBR-RL: Energy balancing routing protocol based on reinforcement learning for WSN. In Proceedings of the 36th Annual ACM Symposium on Applied Computing, Virtual Event, 22–26 March 2021; pp. 1915–1920.
46. Gibbons, A. *Algorithmic Graph Theory*; Cambridge University Press: New York, NY, USA, 1985; pp. 121–134.
47. Prim, R.C. Shortest connection networks and some generalizations. *Bell Syst. Tech. J.* **1957**, *36*, 1389–1401. [CrossRef]
48. Kruskal, J.B. On the shortest spanning subtree of a graph and the traveling salesman problem. *Proc. Am. Math. Soc.* **1956**, *7*, 48–50. [CrossRef]
49. Halim, Z. Optimizing the minimum spanning tree-based extracted clusters using evolution strategy. *Clust. Comput.* **2018**, *21*, 377–391. [CrossRef]
50. de Almeida, T.A.; Yamakami, A.; Takahashi, M.T. An evolutionary approach to solve minimum spanning tree problem with fuzzy parameters. In Proceedings of the IEEE International Conference on Computational Intelligence for Modelling, Control and Automation and International Conference on Intelligent Agents, Web Technologies and Internet Commerce, Washington, DC, USA, 28–30 November 2005; Volume 2, pp. 203–208.
51. Almeida, T.A.; Souza, V.N.; Prado, F.M.S.; Yamakami, A.; Takahashi, M.T. A genetic algorithm to solve minimum spanning tree problem with fuzzy parameters using possibility measure. In Proceedings of the IEEE NAFIPS Annual Meeting of the North American Fuzzy Information Processing Society, Detroit, MI, USA, 26–28 June 2005; pp. 627–632.
52. Hagberg, A.; Swart, P.; Daniel, S.C. Exploring network structure, dynamics, and function using NetworkX. In Proceedings of the 8th SCIPY Conference, Pasadena, CA, USA, 19–24 August 2008; pp. 11–15.