Centralized Routing for Lifetime Optimization Using Genetic Algorithm and Reinforcement Learning for WSNs

Elvis Obi
Computer Science Research Institute
Paul Sabatier University
Toulouse, France
email: elvis.obi@irit.fr

Zoubir Mammeri
Computer Science Research Institute
Paul Sabatier University
Toulouse, France
email: zoubir.mammeri@irit.fr

Okechukwu E. Ochia
Department of Electrical Engineering
University of Calgary
Calgary, Canada
email: okechukwu.ochia1@ucalgary.ca

***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. This paper discusses an optimization method for routing in WSNs to extend the network lifetime. Although the conventional method can extend the network lifetime, the computation time increases exponentially with the number of sensor nodes. Therefore, this method cannot apply to large-scale WSNs. This paper proposes a method to reduce the computation time using a genetic algorithm and shows that the proposed method can provide a suboptimal routing path through evaluation experiments.**

***Index Terms*—reinforcement learning, genetic algorithm, routing, wireless sensor network, network lifetime, path optimization**

I. Introduction

Wireless Sensor Networks (WSNs) consist of spatially deployed sensor nodes in a geographical area to monitor and/or track physical properties, such as motion, pressure, temperature, etc., to collect, process and communicate the data to a sink using the wireless medium [1]. This has made WSNs useful in different fields of application, such as battlefield monitoring, object tracking, environmental monitoring, disaster management, etc. The sensor nodes have limited resources, such as power, bandwidth, memory, storage, processing, and computing speed. The sensor node is made up of four units, namely: the sensing unit, power unit, processing unit, and communication unit. The power unit is limited in energy source and supplies energy to the other units. Sensor nodes are mostly deployed in harsh environments, which makes their battery replacement difficult [2]. This makes the energy-efficient utilization of the sensor nodes vital to increase the network lifetime [3].

Distributed control with self-organized management has been the way of operating WSNs. This approach consumes a lot of energy due to the control overhead, which emanates from the periodically broadcast messages for topology discovery by each sensor node to discover neighbors within its transmission range. But, recently Software-Defined WSNs (SDWSNs) have been proposed to enable WSNs to utilize the benefits of Software-Defined Networking (SDN). SDN is an emerging technology that intends to overcome the limited resources of sensor nodes and static architectures of networks by decoupling the network control plane from the data plane, bringing about programmability, and grouping the network into the application plane, control plane, and data plane. The network policies for routing, load balancing, etc. are being defined and administered at the application plane. The control plane evaluates and transforms the network policies into routing rules. The generation and forwarding of traffic using the routing rules are carried out in the data plane under the management and supervision of the controller in the control plane. The controller has wholistic information of the network and communicates with the sensor nodes using multi-hop communication [4].

The energy constraint problem in WSN, which determines the network lifetime is therefore alleviated in the SDWSN paradigm by removing some of the energy-consuming functions, such as sending control packets from the sensor nodes to the controller. This makes the sensor nodes not have any intelligence and functions, such as load balancing, routing, etc. to be handled at the controller or/and application layer. Routing in SDWSN is the selection of paths by the controller for the sensor nodes in forwarding data packets to the sink using multihop communication. Since the controller has a wholistic knowledge of the network topology, it can generate possible Minimum Spanning Trees (MSTs) to be used as the routing paths [5].

Traditional SDWSN has a limitation of finding the best routing path(s) such that the residual energy of sensor nodes is balanced, this is because it uses predetermined routing paths for data transmission. This thereby degrades the network lifetime since the predetermined routing path in advance does not depict the real status of the network, for instance, the energy consume by the sensor nodes to send packets to the

 ISBN: 978-1-68558-005-6

sink. This problem of finding an energy-efficient way of using the best routing path(s) in SDWSN can be solved by deploying artificial intelligence, such as Reinforcement Learning (RL) and Genetic Algorithm (GA) at the controller. RL is a kind of machine learning that learns to solve a problem by trial and error [6]. GA is a search-based heuristic technique based on the concept of natural selection and genetics [7]. The GA can be used to remove the NP-hardness of the All MSTs algorithm for a large-scale WSNs. This enables the generation of a subset of the network MSTs which serve as the routing tables in polynomial time. The RL enables the controller to learn the MSTs that maximize the network lifetime.

This paper aim at improving the time and space complexity of the Lifetime-Aware Centralized Q-Routing Protocol (LACQRP) [5]. This is achieved by using the proposed genetic MSTs algorithm to generate the routing tables. Also, the network lifetime considered in LACQRP is extended from the time for the first sensor node to deplete its energy source to the time that the sink is not reachable by the alive sensor nodes. This is because sensing is still possible by the alive sensor nodes and their data can be forwarded to the sink if the network graph is still connected.

The rest of the paper is structured as follows: Section II is the review of similar works; Section III explains the design of the Centralized Routing Protocol for Lifetime Optimization using GA and RL (CRPLOGARL); Section IV provides the discussion of the simulation and results and the conclusion of the paper is given in section V.

II. Review of Similar Works.

The first RL-based routing protocol in networks is called Q-routing by Boyan and Littman [8]. Q-routing is a hop-by-hop routing protocol that minimizes packet delivery delay in networks. The limitations of Q-routing are Q-value freshness, parameter setting sensitivity, and slow convergence. Different routing protocols such as [9] - [14] have also applied RL to optimize delivery delay in networks.

Network lifetime and energy optimization routing protocols based on RL for WSNs have been done by different authors. The sequel presents some of these works.

Zhang and Fromherz [15] proposed an energy-efficient routing protocol based on RL for WSNs called constrained flooding. 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. Dong et al. [16] designed for ultra-wideband sensor networks a RL based Geographical Routing (RLGR) protocol. RLGR improved the network lifetime by uniformly distributing energy consumption among nodes and reducing packet delivery delay. RLGR considers residual energy of nodes and hop counts to the sink in its reward function. Simulation results showed that RLGR has better network lifetime by at least 75 percent with respect to Greedy Perimeter Stateless Routing (GPSR) [17].

Hu and Fei [18] proposed for Underwater WSNs (UWSNs) a Q-learning-based Energy-efficient and Lifetime-Aware Routing (QELAR) protocol for finding the optimal routing path in the network. 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.

Jafarzadeh and Moghaddam [19] proposed a routing protocol for WSNs called Energy-aware QoS Routing RL-based (EQR-RL) protocol. 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 packet sender and the selected forwarder.

Geo et al. [20] proposed an intelligent routing protocol for WSNs built on RL named RL-based Lifetime Optimization (RLLO) routing protocol. RLLO uses residual energy of 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.

Debowski et al. [21] proposed a hybrid protocol called Q-Smart Gradient-based (QSGrd) routing protocol for WSNs. 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 best routing paths are learned and selected based on the residual energy of the next hop and the average least number of transmissions to the sink using RL.

Mutombo et al. [22] proposed an RL-based Energy Balancing Routing (EBR-RL) protocol for WSNs. 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.

Obi et al. [5] proposed a Lifetime-Aware Centralized Q-Routing Protocol (LACQRP) for WSNs. The state space and action space of LACQRP are all MSTs generated by the controller which served as the routing tables. LACQRP maximizes the network lifetime by learning the optimal routing tables that minimize the maximum energy consumption of the sensor

 ISBN: 978-1-68558-005-6

nodes. Simulation results show that LACQRP converges to the optimal routing table(s) and has an increased network lifetime when compared with RL-Based Routing (RLBR) [23] and RL for Lifetime Optimization (R2LTO) [24]. However, LACQRP computational complexity varies exponentially with the number of deployed sensor nodes. This makes LACQRP not to be feasible in practice for large-scale WSNs.

III. Methodology

This paper considers a WSN consisting of a set of sensor nodes and a sink. The sink also acts as an SD controller. 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 $\mathcal{V}$ is the set of sensor nodes and $\mathcal{E}$ is the set of network links between two connected distinct 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 controller computes a list of routing tables using the proposed genetic MSTs algorithm based on distance edge weight. The distance edge weight is used to compute the MSTs because the transmission energy of a sensor node depends on distance. The controller chooses an MST in each round using RL and broadcasts it to all sensor nodes for data transmission. This enables the lifetime optimization of the WSN globally. The genetic MSTs algorithm and the centralized routing protocol for lifetime optimization using GA and RL are presented in the sequel.

A. A Genetic MSTs Algorithm

The set of nodes and links of the network graph are denoted by $\mathcal{V} = \{v_1, ..., v_n\}$ and $\mathcal{E} = \{e_1, ..., e_m\} \subseteq \mathcal{V} \times \mathcal{V}$, respectively. Each link $e \in \mathcal{E}$ is associated with an integer weight $w(e) > 0$ representing the link distance. The population of the GA is obtained from MSTs generated by a classical MST algorithm. The classical algorithms for finding MST of a connected undirected graph are Kruskal's algorithm [25], Prim's algorithm [26], and Boruka's algorithm [27]. 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 [28]. Therefore, the number of MSTs generated by the Prim's algorithm by varying the root node is less than or equal to the number of nodes, $|\mathcal{V}|$ of the network graph.

The genetic MSTs algorithm uses unique MSTs of the network as the initial population extracted by the Prim's algorithm. Prim's algorithm is called using the network graph and varying root nodes as inputs. The algorithm for generating the MSTs using Prim's algorithm runs in $O(|\mathcal{V}||\mathcal{E}| \log |\mathcal{V}|)$ time, where $|\mathcal{V}|$ and $|\mathcal{E}|$ denote the number of sensor nodes and network links, respectively. The algorithm for generating the initial population for the genetic MSTs algorithm is given in **Algorithm 1**.

Algorithm 1 Initial population using Prim's Algorithm

Input: $\mathcal{G}(\mathcal{V}, \mathcal{E})$
Output: $MSTs$

```
MSTs = {}
for j in V do
    Select vertex j as the root node
    T = Prim(G, j)
    if T ∉ MSTs then
        MSTs ← T
    end if
end for
Return MSTs
```

Every chromosome in the population represents an MST, with its genes representing the graph edges [29]. 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 (1).

$$fitness(k) = \text{Poss}\left[\sum_{i,j \in T^k} d_{i,j} = \sum_{i,j \in T^*} d_{i,j}\right] \quad (1)$$

where $d_{i,j}$ is a link distance weight.

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 [30]. The new individual generated will be an MST of $\mathcal{G}_1$ and also of the network graph $\mathcal{G}$. The crossover rate, cr determines how many times of applying the crossover operator before taking the fitness individual.

Also, 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, $\mathcal{G}^*_2$ [29]. 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 mutation rate, mr is used to specify the number of times of applying the mutation operator before selecting the fitness one. The genetic MSTs algorithm is as given in **Algorithm 2**.

 ISBN: 978-1-68558-005-6

Algorithm 2 A Genetic MSTs algorithm

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

$P = \{\}$
Generate $k \leq |\mathcal{V}|$ 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**
    Randomly choose T_1, T_2 from 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**
    Randomly choose T_3 from P
    Randomly choose $e_{i,j}$ from T_3
    $\mathcal{G}_2 = T_3 - e_{i,j}$
    $\mathcal{G}^* = \mathcal{G} - e_{i,j}$
    $Cut\,Set = \{e_{i,j} \mid e_{i,j} \in \mathcal{G}^* \;\&\; e_{i,j} \notin \mathcal{G}_2\}$
    Randomly choose $e_{i,j}$ from $Cut\,Set$
    $\mathcal{G}_2^* = \mathcal{G}_2 + e_{i,j}^*$
    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 (1)
    if $fitness$ is True **then**
      if $T \notin P$ **then**
        $P \leftarrow T$
      end if
    end if
  end for
end for
Return P

B. A Centralized Routing Protocol for Lifetime Optimization using Genetic Algorithm and Reinforcement Learning

A Centralized Routing Protocol for Lifetime Optimization using GA and RL (CRPLOGARL) is designed to remove the NP-hardness associated with the LACQRP [5]. This is achieved by replacing the All MSTs algorithm in LACQRP with the proposed genetic MSTs algorithm. The CRPLOG-ARL 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 CRPLOGARL resides in the controller of the SDWSN. The action space A and the state space S of the agent is the list of MSTs generated by the controller using **Algorithm 2**. 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 used. 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 (2) [6]. The learning of the agent is made meaningful by denoting the Q-value of the learning agent as the maximum of the sensor nodes' energies consumption when a particular MST is used in receiving data packets by the sink.

$$Q^{new}(s_t, a_t) \leftarrow (1-\alpha)Q^{old}(s_t, a_t) + \alpha\left[r_t + \gamma * \max_{a \in A}\{Q(s_{t+1}, a)\}\right] \tag{2}$$

The extent to which the new 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 recent 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.

In the proposed protocol, $Q_0(s_0, a_0)$ is initialized as zero. This is because no energy is consumed by the sensor nodes when there is no sending of data packets to the sink. The achievable reward r_t in each learning episode is given in (3).

$$r_t = \max_{v \in V}\{EC_v\} \tag{3}$$

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 (4).

$$EC_v = ESRE_v^{Previous} - ESRE_v^{Current} \tag{4}$$

The energy model adopted for CRPLOGARL is thesame as LACQRP [5]. The reward function is minimized by selecting the MST that has a minimum Q-value using the epsilon-greedy technique [6]. 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 (5).

$$a_t = \begin{cases} \underset{a \in A}{\operatorname{argmin}}\{Q_t(s, a)\},\ if\ r < 1 - \epsilon \\ Random\ action,\ otherwise. \end{cases} \tag{5}$$

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 (6).

$$\zeta_{MST} = \frac{\tau_{MST}}{NE} \tag{6}$$

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 CRPLOGARL for finding the optimal RT 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 3**.

Algorithm 3 CRPLOGARL

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

```
Controller executes Algorithm 2
for i = 1 to L do
    Initialize Q_0(s_0, a_0) = 0.
    Initialize s_0 as a random MST
    Select an MST using (5).
    Broadcast the MST to all sensor nodes.
    Sink receive data from the sensors using the MST.
    Computes the reward using (3).
    Updates Q^new using (2).
    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 step 3 to 14
        else
            break
        end if
    end if
end for
```

The initialized Q-matrix of **Algorithm 3** depends on the number of generated MSTs, N by **Algorithm 2** that are used as routing tables. This makes the time complexity of **Algorithm 3** to be the same as **Algorithm 2**. Likewise, **Algorithm 3** has a space complexity of $O(N)$.

IV. Simulation and Results Discussions

The performance of the proposed genetic MSTs algorithm is first established for convergence using simulations and compared with the All MSTs algorithm [31] using the number of MSTs generated and computation time as the performance indices. Subsequently, the performance of the CRPLOGARL is achieved by simulations using the performance metrics of network lifetime, number of alive sensor nodes (NAN), and computation time. These metrics of the CRPLOGARL are compared with that of the recent LACQRP [5] 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 at the network lifetime. The computation time is the central processing unit (CPU) time taken to achieve the network lifetime.

The CRPLOGARL 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 [32]. 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 CRPLOGARL and the LACQRP are shown in Table 1.

TABLE I
Simulation Parameters

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	50 m
Bandwidth	1 $kbps$
Data packet size	1024 bits
Initial sensor energy	1 J to 10 J
Packet generation rate	$1/s$ to $10/s$
Discount factor	0.0
Learning rate	0.7
Epsilon	0.1
Number of generations	1000
Crossover rate	0.01
Mutation rate	1

As shown in Fig. 1, when the mutation rate is kept constant and the crossover rate is increased for the scenario when the number of network edges is 5000, the genetic algorithm for generating MSTs converges slower. This is because the higher the crossover rate, the lower the number of crossover operations that are performed by the genetic MSTs algorithm.

 ISBN: 978-1-68558-005-6

This leads to a reduced possibility of finding newer individuals in a given generation. There is a direct relationship between the speed of convergence and the crossover rate of the genetic MSTs algorithm. This is because the crossover operation always leads to the formation of an MST of the network graph.

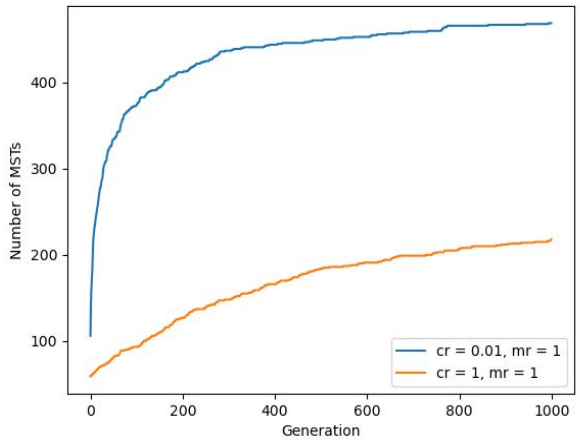

Fig. 1. Number of MSTs in each generation with varied crossover rate.

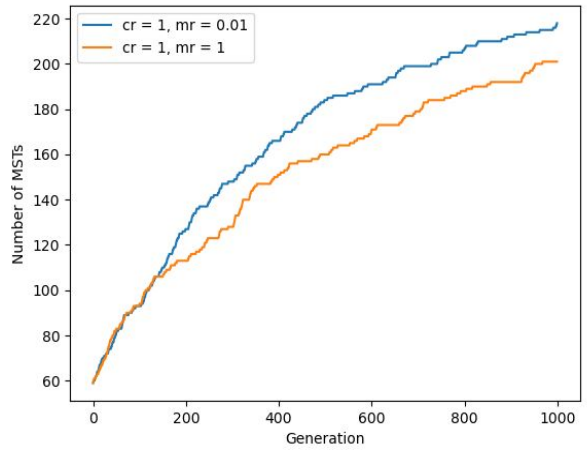

Fig. 2. Number of MSTs in each generation with varied mutation rate.

Consequently, as shown in Fig. 2, when the crossover rate is kept constant and the mutation rate is increased, the lower the number of mutation operations that are performed by the genetic MSTs algorithm. This leads to a reduced possibility of finding newer individuals in a given generation. Though there is no direct relationship between the speed of convergence and the mutation rate of the genetic MSTs algorithm. This is because the mutation operation does not always lead to the formation of an MST of the network graph. The choice of the crossover rate, and mutation rate for the CRPLOGARL is gotten from simulations as shown in Fig. 1 and Fig. 2.

The performance of the proposed genetic MSTs algorithm is ascertained by comparing it with the All MSTs algorithm using the number of generated MSTs and the computation time as the performance metrics. The number of the network nodes is kept constant at 101, while the number of deployed edges is varied from 1000 to 5000 at an interval of 1000. The edges are formed randomly between nodes while ensuring there are no cycles formed in the network graph. The number of MSTs generated and the corresponding computation time of the proposed genetic MSTs algorithm and the All MSTs algorithm are given in Table II.

TABLE II
PERFORMANCE COMPARISON OF PROPOSED GENETIC MSTS ALGORITHM WITH ALL MSTS ALGORITHM

Edges	GA MSTs		All MSTs	
	MSTs	*Time (s)*	*MSTs*	*Time (s)*
1000	51	5.89	54	8.41
2000	103	12.74	168	23.92
3000	296	17.29	504	62.63
4000	345	26.01	1240	131.15
5000	496	41.65	4262	301.45

The number of MSTs generated by the genetic MSTs algorithm and the All MSTs algorithm increases with the number of randomly deployed edges as shown in Table II. The genetic MSTs algorithm can find an average of 20.73% when compared with all MSTs generated by the All MSTs algorithm with a reduced computation time of 80.48%.

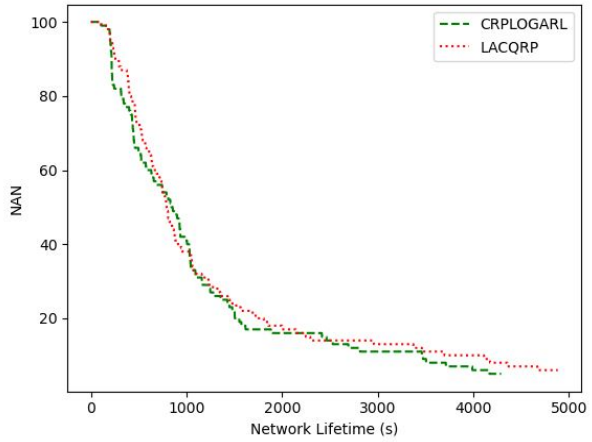

Fig. 3. Number of alive sensors with network lifetime.

Fig. 3 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 energies and packet gemeration rate of the sensor nodes are kept arbitrarily at 1 J and 1 $/s$, respectively. 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. The CRPLOGARL has 5 alive sensor nodes at the network lifetime. This is against the LACQRP with 6 alive sensor nodes before the sink is no longer reachable for data transmission. This resulted to CRPLOGARL having 16.67% degradation in NAN performance when compared with LACQRP. Therefore the CRPLOGARL makes the network graph to be disconnected faster when compared with LACQRP. This is because CRPLOGARL only uses a

subset of all MSTs that does not contain all the optimal ones. This led to a reduced network lifetime in CRPLOGARL when compared with LACQRP.

Subsequently, the network lifetime with increasing the initial sensor nodes energies for both routing protocols is as shown in Fig. 4. As the initial sensor nodes energies increases, the network lifetime of both routing protocols increases. This is because the network lifetime is proportional to the residual energies of the sensor nodes. The CRPLOGARL has a lower network lifetime performance of 9.88% when compared with LACQRP. This is because of LACQRP generates all MSTs which include the optimal ones and as the initial node energies of the sensor nodes increases, the LACQRP tends to use often the optimal MSTs which led to better network lifetime. However, due to the NP-hardness of generating all MSTs in LACQRP, CRPLOGARL has reduced computation time of 90.87% when compared with LACQRP has shown in Fig. 5.

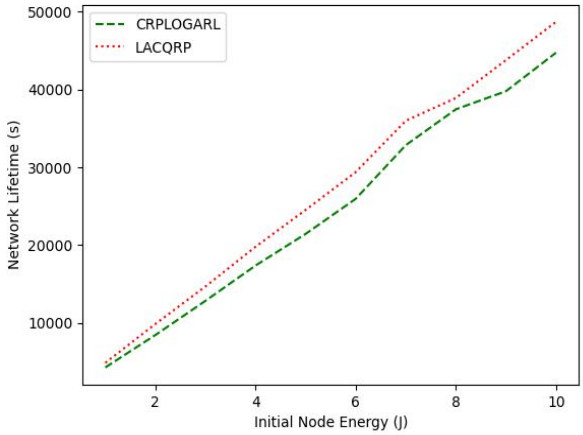

Fig. 4. Network lifetime with initial sensor energy.

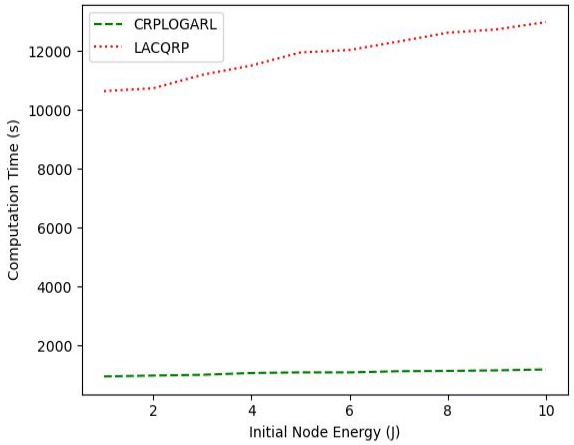

Fig. 5. Computation time with initial sensor energy.

Consequently, the network lifetime with increasing sensors packet generation rate for both routing protocols is as shown in Fig. 6. As the packet generation rate increases, the network lifetime of both routing protocols decreases. This is because the network lifetime is inversely proportional to the packet generation rates of the sensor nodes. The CRPLOGARL has a lower network lifetime performance of 7.92% when compared with LACQRP.

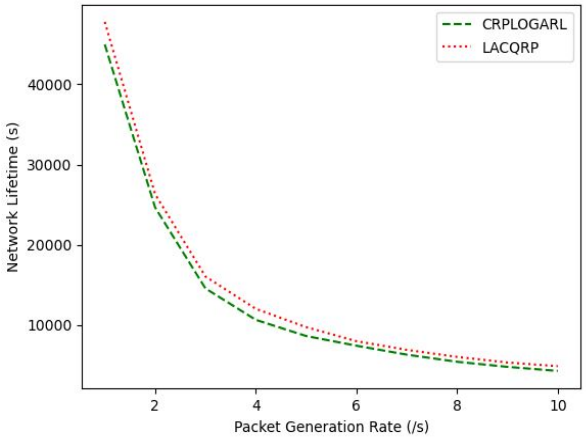

Fig. 6. Network lifetime with rate of packet generation.

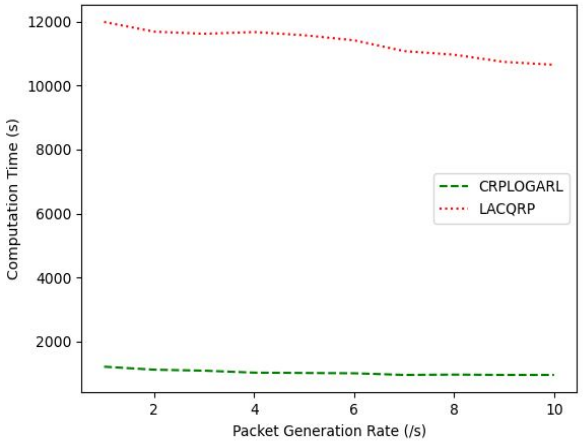

Fig. 7. Computation time with rate of packet generation.

This is because of LACQRP generates all MSTs which include the optimal ones and as the packet generation rates of the sensor nodes increases, the LACQRP tends to use less the optimal MSTs. However, due to the NP-hardness of generating all MSTs in LACQRP, CRPLOGARL has reduced computation time of 90.90% when compared with LACQRP has shown in Fig. 7.

V. Conclusion

This paper presented the design of a centralized routing protocol for lifetime optimization using a genetic algorithm and reinforcement learning for WSNs. The sink and the controller resided at the base station of the Software Defined WSN. The controller generated subsets of all minimum spanning trees of the network graph in polynomial time for routing purposes.

 ISBN: 978-1-68558-005-6

The reinforcement learning deployed at the controller learned the optimal MST(s) for lifetime optimization at each stage of the network building by the controller. This maximized the time taken for the sink not to be reachable by alive sensor nodes. The centralized routing protocol for lifetime optimization using genetic algorithm and reinforcement learning has improved computation time when compared with the lifetime-aware centralized routing protocol. This enabled the real-life implementation of the proposed protocol feasible. However, the proposed protocol suffered a reduction in the performance of network lifetime and the number of alive nodes when the network graph is disconnected. This is because the sub-set of the MSTs generated by the proposed protocol does not contain all the optimal MST(s). Future work will consider emulating the Software Defined WSN using Mininet considering real-world parameters of a typical WSN.

ACKNOWLEDGMENT

Elvis Obi was sponsored by the Petroleum Technology Trust Fund (PTDF) Overseas Scholarship Scheme of Nigeria government.

REFERENCES

[1] D. S. Ibrahim, A. F. Mahdi, and Q. M Yas, "Challenges and Issues for Wireless Sensor Networks: A Survey," Journal of Global Scientific Research, vol. 6, no. 1, pp. 1079-1097, January 2021.

[2] R. Priyadarshi, B. Gupta, and A. Anurag, "Deployment techniques in wireless sensor networks: a survey, classification, challenges, and future research issues," The Journal of Supercomputing, vol. 76, no. 9, pp. 7333-7373, January 2020.

[3] Z. Mammeri, "Reinforcement learning based routing in networks: Review and classification of approaches," IEEE Access, vol. 7, pp. 55916-55950, April 2019.

[4] M. Ndiaye, G. P. Hancke, and A. M. Abu-Mahfouz, "Software Defined Networking for Improved Wireless sensor Network Management: A Survey," Sensors, vol. 17, no. 5, pp. 1-32, May 2017.

[5] E. Obi, Z. Mammeri, and O.E. Ochia, "A Lifetime-Aware Centralized Routing Protocol for Wireless Sensor Networks using Reinforcement Learning," In 17th IEEE International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob), October 2021, pp. 363-368.

[6] R. S. Sutton and A. G. Barto, "Reinforcement Learning: An introduction," 2nd ed., Cambridge, MA, USA: MIT press, 2018.

[7] D. A Whitley, "Genetic algorithm tutorial," Statistics and computing, vol. 4, no. 2, pp. 65-85, June 1994.

[8] J. A. Boyan and M. L. Littman, "Packet routing in dynamically changing networks: A reinforcement learning approach," In Advances in neural information processing systems, vol. 6 pp. 671-678, 1993.

[9] S. P. Choi and D. Y. Yeung, "Predictive Q-routing: A memory-based reinforcement learning approach to adaptive control," in 9th Proceedings of Neural Information Processing Systems Conference, 1996, pp. 946–951.

[10] D. Subramanian, P. Druschel, and J. Chen, "Ants and reinforcement learning: A case study in routing in dynamic networks," in 15th Proceedings of International Joint Artificial Intelligence Conference, 1997, pp. 832–839.

[11] S. Kumar and R. Miikkulainen, "Confidence based Q-routing: An online network routing algorithm," in 16th Proceedings of Artificial Neural Network Engineering Conference, San Francisco, CA, USA, 1998, pp. 758–763.

[12] L. Peshkin and V. Savova, "Reinforcement learning for adaptive routing," in Proceedings of International Joint Conference on Neural Networks, 2002, pp. 1825–1830.

[13] D. Chetret, C. K. Tham, and L. Wong, "Reinforcement learning and CMAC-based adaptive routing for MANETs," in 12th Proceedings of IEEE International Conference on Networks, November 2004, pp. 540–544.

[14] P. Fu, J. Li, and D. Zhang, "Heuristic and distributed QoS route discovery for mobile ad hoc networks," in 5th Proceedings of International Conference on Computer and Information Technology, Shanghai, China, 2005, pp. 512–516.

[15] Y. Zhang, and M. Fromherz, "Constrained flooding: a robust and efficient routing framework for wireless sensor networks," In 20th IEEE International Conference on Advanced Information Networking and Applications, May 2006, pp. 1-6.

[16] S. Dong, P. Agrawal, and K. Sivalingam, "Reinforcement learning based geographic routing protocol for UWB wireless sensor network," In IEEE Global Telecommunications Conference, November 2007, pp. 652-656.

[17] B. Karp and H.T. Kung, "GPSR: Greedy perimeter stateless routing for wireless networks," In 6th annual international conference on Mobile computing and networking, August 2000, pp. 243-254.

[18] T. Hu and Y. Fei, "QELAR: A machine-learning-based adaptive routing protocol for energy-efficient and lifetime-extended underwater sensor networks," IEEE Transactions on Mobile Computing, vol. 9, no. 6, pp. 796–809, February 2010.

[19] S. Z. Jafarzadeh and M. H. Y. Moghaddam, "Design of energy-aware QoS routing algorithm in wireless sensor networks using reinforcement learning," In 4th IEEE International Conference on Computer and Knowledge Engineering, May 2014 pp. 722-727.

[20] W.J. Guo, C.R. Yan, Y.L. Gan, and T. Lu, "An intelligent routing algorithm in wireless sensor networks based on reinforcement learning," Applied Mechanics and Materials, Vol. 678, pp. 487–493, October 2014.

[21] B. Debowski, P. Spachos, and S. Areibi, "Q-learning enhanced gradient based routing for balancing energy consumption in WSNs," In 21st IEEE International Workshop on Computer Aided Modelling and Design of Communication Links and Networks, December 2016 pp. 18-23.

[22] V. K. Mutombo, S. Y. Shin, and J. Hong, "EBR-RL: energy balancing routing protocol based on reinforcement learning for WSN," In 36th ACM Annual Symposium on Applied Computing Proceedings, April 2021, pp. 1915-1920.

[23] W. Guo, C. Yan, and T. Lu, "Optimizing the lifetime of wireless sensor networks via reinforcement-learning-based routing," International Journal of Distributed Sensor Networks, vol. 15, no. 2, pp. 1-20, February 2019.

[24] S. E. Bouzid, Y. Serrestou, K. Raoof, and M. N. Omri, "Efficient routing protocol for wireless sensor network based on reinforcement learning," In 5th IEEE International Conference on Advanced Technologies for Signal and Image Processing, October 2020, pp. 1-5.

[25] J. B. Kruskal, "On the shortest spanning subtree of a graph and the traveling salesman problem," In Proceedings of the American Mathematical society, vol. 7, no. 1, pp. 48-50, 1956.

[26] R. C. Prim, "Shortest connection networks and some generalizations," The Bell System Technical Journal, vol. 36, no. 6, pp. 1389-1401, 1957.

[27] J. Eisner. "State-of-the-art algorithms for minimum spanning trees: A tutorial discussion", University of Pennsylvania, 1997.

[28] Z. Halim, "Optimizing the minimum spanning tree-based extracted clusters using evolution strategy," Cluster Computing, vol.21, no.1, pp. 377-391, March 2018.

[29] T. A. Almeida, V. N. Souza, F. M. S. Prado, A. Yamakami, and M. T. Takahashi, "Genetic algorithm to solve minimum spanning tree problem with fuzzy parameters using possibility measure," In IEEE NAFIPS Annual Meeting of the North American Fuzzy Information Processing Society, pp. 627-632, June 2005.

[30] T. A. Almeida, A. Yamakami, and M. T. Takahashi, "An evolutionary approach to solve minimum spanning tree problem with fuzzy parameters," In 6th IEEE International Conference on Computational Intelligence for Modelling, Control and Automation and International Conference on Intelligent Agents, Web Technologies and Internet Commerce, Vol. 2, November 2005, pp. 203-208.

[31] T. Yamada, S. Kataoka, and K. Watanabe, "Listing all the minimum spanning trees in an undirected graph," International Journal of Computer Mathematics, vol. 87 no. 14, pp. 3175-3185, November 2010.

[32] A. Hagberg, P. Swart, and S. C. Daniel, "Exploring network structure, dynamics, and function using NetworkX, In 8th SCIPY Conference, August 2008, pp. 11-15.

 ISBN: 978-1-68558-005-6