DQR: A Double Q Learning Multi Agent Routing Protocol for Wireless Medical Sensor Network

Muhammad Shadi Hajar[1(✉)], Harsha Kalutarage[1], and M. Omar Al-Kadri[2]

[1] Robert Gordon University, Aberdeen AB10 7GJ, UK
{m.hajar,h.kalutarage}@rgu.ac.uk
[2] Birmingham City University, Birmingham B4 7XG, UK
omar.alkadri@bcu.ac.uk

Abstract. Wireless Medical Sensor Network (WMSN) offers innovative solutions in the healthcare domain. It alleviates the patients' everyday life difficulties and supports the already overloaded medical staff with continuous monitoring tools. However, widespread adoption of these advancements is still restrained by security concerns and limitations of existing routing protocols. Routing is challenging in WMSN owing to the fact that some critical requirements, such as reliable delivery, have been neglected. To address these challenges, this paper proposes DQR, a double Q-learning routing protocol to meet WMSN requirements and overcome the positive bias estimation problem of the Q-learning based routing protocols. DQR uses a novel Reinforcement Learning (RL) model to reduce computational and communication overheads. It is combined with an effective trust management system to ensure a reliable data transfer and defeat packet dropping attacks. The experimental results demonstrate robust performance under various attacks with minimal resource footprint and efficient energy consumption.

Keywords: Double Q-learning · Routing · Reinforcement Learning · Trust management · Blackhole attack · Selective forwarding attack · Sinkhole attack

1 Introduction

Wireless Medical Sensor Network (WMSN) has become a critical element in the healthcare systems to monitor the physiological signs of the human body. This revolutionized technology provides medical staff with continuous real-time monitoring data without disturbing the patients. However, the widespread uptake of WMSN applications is still suppressed by security concerns. Ensuring a secure and reliable data transfer between the sensing units and the sink is still challenging despite the abundant routing protocols proposed for Wireless Sensor Network (WSN) [1,2]. Although WMSN is regarded as a branch of WSN, routing

F. Li et al. (Eds.): SecureComm 2022, LNICST 462, pp. 611–629, 2023.
https://doi.org/10.1007/978-3-031-25538-0_32

protocols and security countermeasures proposed for WSN do not necessarily fit WMSN due to its resource limitations, critical applications, and operating conditions.

Reinforcement Learning (RL) has been used recently to solve distributed optimization problems, such as routing [3]. RL-based routing protocols rely on an existence of a learning agent that acts with the environment and receives rewards based on its actions. By interacting with the network environment, the learning agents will be able to maximize their reward by making optimal forwarding decisions. Q-learning, which is a model-free RL algorithm, is the most used algorithm for both centralized and decentralized routing protocols [4]. Although this approach is able to produce an efficient routing protocol that can outperform other algorithms, it still has drawbacks. First, as it works without prior knowledge about the environment, it requires a series of randomly chosen actions to explore the environment before converging on the optimal solution. WMSN cannot tolerate a long learning period because of its sensitive applications. Second, Q-learning has an inborn overestimation problem which has been overlooked for a long time [5]. It uses the maximum value as an estimation for the maximum expected value. The routing performance may be impacted negatively due to this positive bias. Third, although different parameters have been considered in protocol design, ensuring reliable data transfer is still challenging as senders cannot predict the behaviour of other nodes in the path to the destination. Moreover, taking into consideration more parameters may optimize the routing decisions, but it involves a significant overhead increase, especially when information must be exchanged between learning agents. Therefore, a suitable solution is needed to overcome these aforementioned shortcomings.

The main contribution of this paper is threefold. First, the unique requirements for an efficient, lightweight and reliable routing protocol for WMSN are specified. Second, a double Q-learning trust-aware routing protocol for WMSN has been proposed. Third, extensive analysis has been carried out to ensure the robustness of our proposed protocol under different scenarios.

The rest of this article is organized into six sections as follows. Related work is given in Sect. 2. Section 3 overviews WMSN. DQR routing protocol is described in Sect. 4, followed by evaluation and performance results in Sect. 5. Finally, Sect. 6 concludes this article.

2 Related Work

Developing a secure, reliable and efficient routing protocol for WSN is still an open area of research, and it is more challenging in WMSN due to its resource scarcity and critical applications. Abundant research has been carried out to propose an efficient routing protocol using different metrics and methods. Recently, reinforcement learning has been widely used to find the optimal routing path with minimal overhead. Q-learning, which uses temporal difference (TD) to estimate the value of an action in a given state, is extensively used to build an efficient routing policy. However, Q-leaning suffers from an overestimation problem, which overlooks the optimal action in some cases [5]. Therefore, double

Q-learning, which is an off-policy RL algorithm, is introduced to solve the overestimation problem by using double estimators to approximate the maximum expected value. To the best of our knowledge, only a few works used double Q-learning to develop a routing protocol. Authors in [6] proposed DQLR, a double Q-learning routing protocol for Delay Tolerant Networks (DTN). However, DQLR only used the number of hops between the source and the destination as a metric. It achieved an acceptable delivery ratio under normal operation. However, considering the hop count as the only metric is insufficient to deal with complicated scenarios.

On the other hand, researchers use various metrics to build the Q-learning reward function in order to achieve an efficient routing protocol, such as delivery delay, the number of hops, remaining energy and location information [3,7–9]. Although this kind of metrics could produce an efficient forwarding method, it cannot deal with malicious activities launched by insiders. Therefore, the routing protocol needs a different source of information to make an informed routing decision, such as Trust Management System (TMS). According to our literature review, only two routing protocols proposed integrating a TMS with Q-learning. Authors in [10] proposed a resource and security efficient routing protocol combined with a trust mechanism for WSN. However, this protocol is not reproducible due to missing some details. In [11], the authors integrate the beta distribution based trust scheme with the Q-learning algorithm to achieve a reliable routing protocol for WMSN. However, positioning information is to be periodically provided in order to choose the optimal path, which could not be practical for WMSN. Moreover, it needs further investigation under different packet dropping attacks.

3 Wireless Medical Sensor Network

With the rapid advancement of the low power and intelligent biomedical SNs, WMSN emerged as a special kind of WSN for healthcare applications. It consists of a set of tiny SNs that are distributed inside or offside the body to monitor the body's biosignals. This revolutionized technology empowers physicians to timely monitor their patients and intervene when necessary.

3.1 Network Model

This study assumes a WMSN of a ward in a field hospital as shown in Fig. 1. Due to the ongoing COVID-19 pandemic, field hospitals have become prevalent, especially in developing countries. The ward dimensions are $10\,\text{m} \times 50\,\text{m}$, where a number of hospital beds are distributed efficiently to provide the necessary care and save physical space. A network of 64 SNs was used to simulate the WMSN conforming to IEEE 802.15.6 [12]. The SNs have been distributed randomly across the hospital ward. The topology is star, with one SN acting as a sink. All sensed data is transmitted to the sink, which in turn forwards it to the medical server. The communication range is $5m$. Thus, nodes need to cooperate and relay packets for other adjacent nodes.

3.2 Threat Model

The critical applications of WMSN necessitate a reliable routing protocol as dropped packets may carry sensitive information. Dropping attacks, such as blackhole and selective forwarding attacks, may not just disrupt the network operation but endanger the patient's life. This kind of attack is difficult to deal with as malicious nodes are usually legitimate nodes that pass cryptographic security countermeasures, such as authentication. Dropping attacks have various patterns and may happen for different reasons. An SN could get compromised and stop relaying packets for other nodes intentionally. Even benign nodes could act selfishly to save resources or could get overloaded by an inefficient routing protocol. Therefore, a reliable, efficient, lightweight routing protocol for WMSN that ensures secure data delivery between the sensing units and the sink is required. DQR assumes that all SNs nodes are mutually authenticated and have a copy of the security keys to ensure a high level of secure communication.

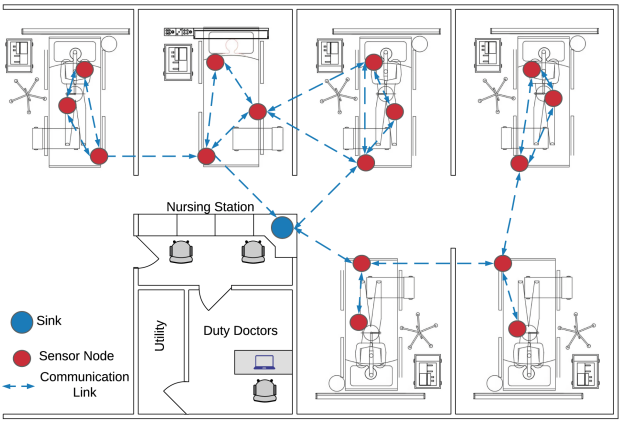


Fig. 1. Network model

4 Protocol Design

In this section, the proposed routing protocol is presented. The design requirements are justified and the proposed algorithms are comprehensively discussed.

4.1 Reinforcement Learning and Double Q-Learning

Multi-Agent Reinforcement Learning (MARL) is a subfield in RL that focuses on studying the behaviour of multiple agents co-existing in a shared environment. The agents are motivated by reward functions and interact with the environment and each other to compete or achieve a common goal. MARL is modeled using the Markov Decision Process (MDP), where the environment has a set of states

$s_t \in \mathbb{S}$ and each agent takes action $a_t \in \mathbb{A}$. In the network environment, each learning agent solves a multi objectives routing problem to make optimal routing decisions.

Q-Learning is an off-policy, model-free temporal difference (TD) algorithm to learn the value of an action in a particular state. However, due to its inborn overestimation problem, Q-Learning could perform poorly in some stochastic environments because the most optimal action could be obscured by overestimation [5]. Therefore, double Q-learning is introduced as an alternative method to approximate the maximum expected action-value by using double estimators.

4.2 Design Requirements

The unique characteristics of WMSN dictate rigorous requirements, which must be kept in mind when designing any potential routing protocol. Therefore, the proposed protocol must be efficient, lightweight, and attack-resistant.

The routing protocol must always choose the optimal path in order to achieve a high delivery ratio and low energy consumption. However, Q-learning-based routing protocols could suffer from poor performance due to the action-value overestimation problem. This biased estimation leads to bad routing decisions that negatively affect the packet delivery ratio. Moreover, increasing the number of transmissions aggravates the energy consumption as transmission activities account for around 80% of the total consumed energy [13]. Therefore, a new approach to achieve an efficient routing protocol is required.

WMSN has stringent resource constraints that make the inherited WSN routing methods not necessarily fit. The traditional RL model necessitates updating the Q table after each sent or forwarded packet, which is a resource depletion process [6,11]. Therefore, the routing engine of any proposed routing protocol must have a minimal resource footprint in terms of processing and memory.

Dropping attacks, such as blackhole and selective forwarding attacks, degrade the overall performance, and most importantly, it may endanger the patient's life. Moreover, the routing process itself could be prone to a specific kind of attack based on the used method, such as poisoning attacks. Therefore, WMSN requires a reliable and robust routing design. The delivery reliability allows the protocol to predict the malicious paths and avoid them, while the design robustness ensures high resiliency to routing attacks.

4.3 DQR Protocol

DQR is designed to fulfill all the above requirements. The reward function is defined as punishment to ensure that the learning agent always chooses the lowest-cost path. Moreover, in order to reduce the computational overhead of the traditional RL model, DQR reformulated the RL model, assuming that the network will be static for a short period, which is an acceptable assumption as nodes could be regarded as stationary for a short interval. This assumption allows the learning agent to perform the same action multiple times before receiving the corresponding reward, as illustrated in Fig. 2. Adopting this method reduces

the computational overhead significantly by updating the Q tables periodically, which will be discussed comprehensively in Sect. 4.4. Furthermore, DQR incorporates an effective TMS to ensure reliable data transfer and avoid malicious paths.

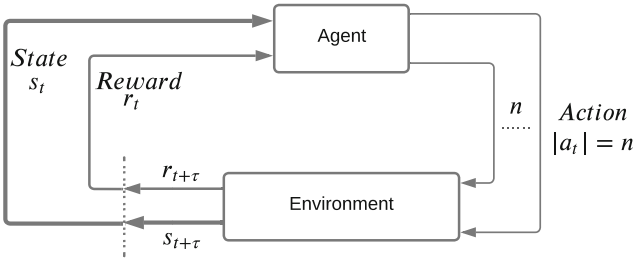


Fig. 2. Graphical representation of the proposed RL model

WMSN network represents the environment $\mathbb{E}$, which contains a set of SNs, one of which acts as a sink S. The learning agent in DQR is defined as the tuple $(\mathbb{S}, \mathbb{A}, \mathbb{R})$ where $\mathbb{S}$ represents a set of states, $\mathbb{A}$ is the set of actions the agent can take, and $\mathbb{R}$ is the reward function. At time step t, an agent at state $s \in \mathbb{S}$ could get a packet to send to destination d, and hence the agent takes action $a_t \in \mathbb{A}$ to forward the packet to one of its neighbors. The learning agent keeps taking the same action during the time window $[t, t+\tau]$. At the end of the time window, the learning agent receives $r_{t+1} \in \mathbb{R}$ from the environment and moves from state s_t to s_{t+1}. DQR defines two Q functions $Q^{A(i)}_{t+1}(s^{(i)}_t, a^{(i)}_t)$ and $Q^{B(i)}_{t+1}(s^{(i)}_t, a^{(i)}_t)$ as the estimated future reward of agent i at state s_t taking the action a_t as shown in Eq. 2 and Eq. 4. Each one of these estimators is updated using a value from the other estimator for the next state as shown in Eq. 1 and Eq. 2. Therefore, the actions a^*_t and b^*_t are the maximum valued actions for $Q^{A(i)}_{t+1}(s^{(i)}_t, a^{(i)}_t)$ and $Q^{B(i)}_{t+1}(s^{(i)}_t, a^{(i)}_t)$, respectively.

$$a^{*(i)}_t = \underset{a^{(i)}_t \in A}{argmax} Q^{A(i)}_t(s^{(i)}_{t+1}, a^{(i)}_t) \tag{1}$$

$$Q^{A(i)}_{t+1}(s^{(i)}_t, a^{(i)}_t) \leftarrow (1-\eta) Q^{A(i)}_t(s^{(i)}_t, a^{(i)}_t) + \eta[r^{(i)}_{t+1}(s^{(i)}_{t+1}) + \gamma Q^{B(i)}_{t+1}(s^{(i)}_{t+1}, a^{*(i)}_t)] \tag{2}$$

$$b^{*(i)}_t = \underset{a^{(i)}_t \in A}{argmax} Q^{B(i)}_t(s^{(i)}_{t+1}, a^{(i)}_t) \tag{3}$$

$$Q^{B(i)}_{t+1}(s^{(i)}_t, a^{(i)}_t) \leftarrow (1-\eta) Q^{B(i)}_t(s^{(i)}_t, a^{(i)}_t) + \eta[r^{(i)}_{t+1}(s^{(i)}_{t+1}) + \gamma Q^{A(i)}_{t+1}(s^{(i)}_{t+1}, b^{*(i)}_t)] \tag{4}$$

where $\eta \in [0, 1]$ is the learning parameter where small values decelerate the learning and large ones may prevent algorithm convergence, $\gamma \in [0, 1]$ is the future reward discount parameter where small values make the learning agent nearsighted by considering the only immediate reward.

DQR is designed to always choose the most reliable shortest path by defining the reward function as punishment, as shown in Eq. 5. The delivery reliability is achieved by incorporating trust information, which is discussed in Sect. 4.6, while the punishment design reduces the number of transmissions along the path to the destination to ensure an energy-efficient protocol. Moreover, energy information from the agent itself is also considered to optimize the network lifetime, which will be discussed further in Sect. 4.5

$$r_{t+1}^{(i)}(s_{t+1}^{(i)}, j) = \begin{cases} -(1 - T_t^{(ij)}).F_t^{(i)} & if \quad O_t^{(ij)} \neq \{\phi\} \\ -(1 - T_{t-\delta}^{(ij)}).F_t^{(i)} & if \quad O_t^{(ij)} = \{\phi\} \wedge |O^{(ij)}| > \epsilon \\ 0 & Otherwise \end{cases} \tag{5}$$

where $r_{t+1}^{(i)}(s_{t+1}^{(i)}, j)$ is the received reward by node i for taking the action $a_t^{(i)} = j$ forwarding the traffic to the neighbor j at time window $[t, t+\tau]$, $T_t^{(ij)}$ is the trust value of SN j that maintained by SN i at time window t and is evaluated using Algorithm 4, $O_t^{(ij)}$ is the direct observations maintained by node i for node j at time window t, δ is a time lag used to obtain the last trust value, ϵ is a threshold to identify the minimum required evidence where higher values means more historical data is required to use the evaluated trust value.

The learning process must be continual due to network dynamicity and distributed as no agent has a full view of the network. DQR is a decentralized protocol where the learning agents exchange their best estimations with their neighbors, as illustrated in Algorithm 1. The received estimations are then used to update the Q^A and Q^B tables and specify the most optimal next hop. As the goal of the learning agent is to maximize the received reward in the long run, greedy action should not always be taken as routing task is a continual online task and exploiting the greedy action all the time prevents the convergence to the global optimum. Therefore, DQR uses ε-greedy method to balance between exploration and exploitation. The learning agent explores the environment with a probability of θ and exploits it with a probability of $(1 - \theta)$. Initially, the learning agents have no evidence from the network; hence their Q values are initialized to zeros, which is more practical to motivate the agents to explore the environment and does not require any hardware or positioning information like in [11,14].

4.4 Synchronous and Asynchronous Updating

DQR adopted a synchronous Q tables updating method with a view to producing a lightweight routing protocol. Each action-value function is updated with the outcome of the other action-value function as shown in Algorithm 2. The actions $a^{*(i)}$ and $b^{*(i)}$ are the maximum value action in state s_{t+1} for $Q^{A(i)}$ and $Q^{B(i)}$, respectively. Therefore, both Q tables are updated for the same problem but with a different set of evidence to produce an unbiased estimate for all action-value(s). Although the obtained experience is divided between two action-value functions, the algorithm is still data-efficient as selecting the optimal action is computed

Algorithm 1: Routing Protocol

Input:
The reward: $r_{t+1}^{(i)}(s_{t+1}^{(i)}, j)$
The Q tables: Q_t^A & Q_t^B
The trust table: T_t
Output: Optimal next hop $a_t^{(i)}$
Initialization:

$$Q_0^{A(i)}(n^{(i)} \in N_t^{(i)}) = Q_0^{B(i)} = \begin{cases} 0 & if \quad n^{(i)} \neq S \\ 1 & if \quad n^{(i)} = S \end{cases} \qquad // \ N_t^{(i)} \text{ is the adjacent nodes of } i$$

$$T_0^{(i)}(n^{(i)} \in N_t^{(i)}) = E[uni(0,1)] = 0.5$$

$$a_1^{(i)} = \begin{cases} S & if \quad S \in N_1^{(i)} \\ n^{(i)} & | \quad n^{(i)} \in N_1^{(i)} \end{cases} \tag{6}$$

while *TRUE* **do**
  $Wait \quad \tau$
  $Broadcast \quad max(Q_t^{A(i)})$ & $max(Q_t^{B(i)})$
  if $\varepsilon - greedy > \theta$ **then**
    $a_t^{(i)} = \underset{a \in A}{argmax}(\frac{Q_t^{A(i)}(s,.) + Q_t^{B(i)}(s,.)}{2})$
    Calculate $r_{t+1}^{(i)}(s_{t+1}^{(i)}, a_t^{(i)})$ as in Eq. 5
    $Q_{t+1}^{A(i)}(s_t^{(i)}, a_t^{(i)})$ & $Q_{t+1}^{B(i)}(s_t^{(i)}, a_t^{(i)})$ Synchronous update as in algorithm 2
  else
    $a_t^{(i)} \leftarrow n_t^{(i)} \mid n_t^{(i)} \in N_t^{(i)}$
    Calculate $r_{t+1}^{(i)}(s_{t+1}^{(i)}, a_t^{(i)})$ as in Eq. 5
    $Q_{t+1}^{A(i)}(s_t^{(i)}, a_t^{(i)})$ & $Q_{t+1}^{B(i)}(s_t^{(i)}, a_t^{(i)})$ Synchronous update as in algorithm 2
  end
  $s_t^{(i)} \leftarrow s_{t+1}^{(i)}$
end

based on the average Q tables as illustrated in Algorithm 1. As the learning agents collaborate with each other by broadcasting their best estimation to a destination, this information is then used to keep the Q tables updated. However, the learning agent forwards the traffic to only one adjacent node during the time window t, and thus it can only calculate the reward for this action. For instance, node i take the action $a_t^{(i)} = j$ during the time window t and receives two updates from nodes j and k. Consequently, DQR updates the action-value of j with the calculated reward using double Q-learning, while it checks if there is enough evidence about node k to update each action-value separately using Q-learning or keep it unchanged in case of not enough evidence. This method allows DQR to react quickly to any environment change, and at the same time, it immunizes DQR against utilizing false updates from malicious nodes.

On the other hand, although the synchronous updating method is computationally efficient, it may decelerate the convergence as the learning agent, especially in the exploration phase, may make wrong decisions, and thus keep forwarding the traffic to the wrong next hop. In traditional RL mode, the learning agent risks losing one packet each time to update the Q tables. However, by using only synchronous updating, more packets may be lost before updating the Q tables. This usually happens when loops occur. Therefore, DQR uses an

Algorithm 2: Synchronous Updating

Input:
The Q Table: $Q_t^{A(i)}$ and $Q_t^{B(i)}$
The reward: $r_{t+1}^{(i)}(s_{t+1}^{(i)}, j)$
The trust table: T_t
Output: $Q_{t+1}^{A(i)}$ and $Q_{t+1}^{B(i)}$
while $TRUE$ **do**
  $Wait \quad \tau$
  foreach $j \in N_t^i$ **do**
    if $j == a_t^i$ **then**
      $\rho \leftarrow rand(0, 1)$
      if $\rho > 0.5$ **then**
        Define $a^{*(i)} = \underset{a \in A}{argmax} Q_t^{A(i)}(s_{t+1}^{(i)}, a_t^{(i)})$
        $Q_{t+1}^{A(i)}(s_t^{(i)}, a_t^{(i)}) \leftarrow$
          $(1-\eta)Q_t^{A(i)}(s_t^{(i)}, a_t^{(i)}) + \eta[r_{t+1}^{(i)}(s_{t+1}^{(i)}) + \gamma Q_{t+1}^{B(i)}(s_{t+1}^{(i)}, a_t^{*(i)})]$
      else
        Define $b^{*(i)} = \underset{a \in A}{argmax} Q_t^{B(i)}(s_{t+1}^{(i)}, a_t^{(i)})$
        $Q_{t+1}^{B(i)}(s_t^{(i)}, a_t^{(i)}) \leftarrow$
          $(1-\eta)Q_t^{B(i)}(s_t^{(i)}, a_t^{(i)}) + \eta[r_{t+1}^{(i)}(s_{t+1}^{(i)}) + \gamma Q_{t+1}^{A(i)}(s_{t+1}^{(i)}, b_t^{*(i)})]$
      end
    else
      if $|O^{ij}| > \epsilon$ **then**
        $Q_{t+1}^{A(i)}(s_t^{(i)}, j) \leftarrow$
          $(1-\eta)Q_t^{A(i)}(s_t^{(i)}, j) + \eta[r_{t-\delta}^{(i)}(s_{t-\delta}^{(i)}, j) + \gamma \max_{j \in N_t^{(i)}} Q_t^{A(i)}(s_{t+1}^{(i)}, j)]$
        $Q_{t+1}^{B(i)}(s_t^{(i)}, j) \leftarrow$
          $(1-\eta)Q_t^{B(i)}(s_t^{(i)}, j) + \eta[r_{t-\delta}^{(i)}(s_{t-\delta}^{(i)}, j) + \gamma \max_{j \in N_t^{(i)}} Q_t^{B(i)}(s_{t+1}^{(i)}, j)]$
      else
        $Q_{t+1}^{A(ij)} \leftarrow Q_t^{A(ij)}$
        $Q_{t+1}^{B(bij)} \leftarrow Q_t^{B(ij)}$
      end
    end
  end
end

asynchronous updating method to step up the learning process and makes the algorithm converge swiftly. Once a loop is detected or expected, such as when forwarding the packet to its source again, the asynchronous updating method is triggered to penalize both corresponding action-value(s) and allow the learning agent to take the appropriate action accordingly, as detailed in Algorithm 3.

4.5 Energy Model

Optimizing the network lifetime is still a challenging concern in WSN and WMSN in particular. Due to the critical applications of WMSN, dead nodes may have catastrophic consequences. Moreover, in some cases, replacing the battery may need surgical intervention. Considering the residual energy of the adjacent nodes is widely used to maximize the overall network lifetime [15,16]. However, exchanging energy information between adjacent nodes is neither energy nor

Algorithm 3: Asynchronous Updating

Input: A packet to forward: $P_t^{(sd)}$
Output: Updated Routing
while *TRUE* **do**
  if $\forall\, i \in \mathbb{N}$ *receives* $P_{t+\delta}^{(id)}$ **then** // $P_{t+\delta}^{(id)}$ is a packet from i to d after time lag δ
    if $\eta == 1$ **then**
      $r_{t+1}^{(i)}(s_{t+1}^{(i)}, j) = -e^{\eta}(1 - T_t^{(ij)}).F_t^{(i)}$
    else
      $r_{t+1}^{(i)}(s_{t+1}^{(i)}, j) = -(1 - T_t^{(ij)}).F_t^{(i)}$
    end
    if $RQ_{t-1}^{A(i)}(s_{t-1}^{(i)}, j) \wedge RQ_{t-1}^{B(i)}(s_{t-1}^{(i)}, j)$ **then** // $RQ_{t-1}^{A(i)}(s_{t-1}^{(i)}, j)$ is the last expected future reward received from j
      update $Q_t^{A(ij)}$ and $Q_t^{B(ij)}$ using $r_{t+1}^{(i)}$, $RQ_{t-1}^{A(i)}$ and $RQ_{t-1}^{B(i)}$
    else // ζ is the loop penalising parameter
      $Q_{t+1}^{(i)}(s_t^{(i)}, a_t^{(i)} = n_j) \leftarrow Q_t^{(ij)} - \zeta$
    end
    $a_t^{(i)} = \underset{n_t^{(i)} \in N_t^{(i)}}{argmax}\left(\frac{Q_t^{A(i)}(s,.) + Q_t^{B(i)}(s,.)}{2}\right)$
    Update $P_t^{(id)}$
    Send $P_t^{(id)}$
  end
  if $\forall\, i \in \mathbb{N}$ *receives* $P_t^{(jd)} \wedge a_t^{(i)} = j$ **then**
    if $\eta == 1$ **then**
      $r_{t+1}^{(i)}(s_{t+1}^{(i)}, j) = -e^{\eta}(1 - T_t^{(ij)}).F_t^{(i)}$
    else
      $r_{t+1}^{(i)}(s_{t+1}^{(i)}, j) = -(1 - T_t^{(ij)}).F_t^{(i)}$
    end
    if $RQ_{t-1}^{A(i)}(s_{t-1}^{(i)}, j) \wedge RQ_{t-1}^{B(i)}(s_{t-1}^{(i)}, j)$ **then**
      update $Q_t^{A(ij)}$ and $Q_t^{B(ij)}$ using $r_{t+1}^{(i)}$, $RQ_{t-1}^{A(i)}$ and $RQ_{t-1}^{B(i)}$
    else
      $Q_{t+1}^{(i)}(s_t^{(i)}, a_t^{(i)} = n_j) \leftarrow Q_t^{(ij)} - \zeta$
    end
    $a_t^{(i)} = \underset{n_t^{(i)} \in N_t^{(i)}}{argmax}\left(\frac{Q_t^{A(i)}(s,.) + Q_t^{B(i)}(s,.)}{2}\right)$
    Forward $P_t^{(jd)}$
  end
end

computational efficient. In contrast, DQR only used local energy information with a view to reducing the computational overhead and avoiding filtering out false second-hand information. Moreover, it uses two sources of energy information with a view to load balancing energy consumption across the network. When the residual energy percentage is greater than a threshold ϑ, this parameter does not contribute in evaluating the consumed energy ratio $E_t^{(i)} \in [0, 1]$ as shown in Eq. 7. In that case, SNs choose the most reliable shortest path, which in turn makes some nodes overloaded due to their trustworthiness and positions. Therefore, DQR defines the energy consumption ratio $C_t^{(i)}$ to evaluate the extra burden incurred by the node due to relaying activities, as shown in Eq. 8. The weighted average of $E_t^{(i)}$ and $C_t^{(i)}$ is calculated in Eq. 9. As integrating the energy

into the reward function may influence the nodes routing decision to choose a malicious path, the energy factor is bounded by $\lambda \in [0, 1]$ as shown in Eq. 10.

$$E_t^{(i)} = \begin{cases} 0 & if \quad \frac{e_{res}(t)}{e_{init}} > \vartheta \\ 1 - \frac{e_{res}(t)}{e_{init}} & Otherwise \end{cases} \tag{7}$$

$$C_t^{(i)} = 1 - \frac{c_n(t)}{c_a(t)} \tag{8}$$

$$\psi_t^{(i)} = \omega E_t^{(i)} + (1 - \omega) C_t^{(i)} \tag{9}$$

$$F_t^{(i)} = e^{\lambda \psi_t^{(i)}} \tag{10}$$

where $e_{res}(t)$ is the remaining energy at time t, e_{init} is the initial energy, ϑ is the residual energy threshold, $c_n(t)$ is the node normal energy consumption rate, $c_a(t)$ is the overall energy consumption rate, ω is the average weight, λ is the bound parameter where $\lambda = 0$ is used to disable the energy module.

4.6 Trust Model

DQR evaluates the trust relationship between the SNs using the Lightweight Trust Management System (LTMS) [17]. LTMS has been chosen for several reasons. It is a lightweight distributed trust scheme designed to fit WMSN requirements. The trust value is evaluated using a novel updating mechanism that can detect packet dropping attacks with different dropping patterns thanks to integrating the slopes b_t and d_t into the beta distribution shape parameters α_t and β_t, which gives more weight to bad activities and makes it difficult to eliminate. As TMSs can be manipulated by intelligent adversaries who launch on-off attacks, LTMS is provided by a protection module that can detect complicated on-off attacks by considering short and long-term trust values to detect repeated dropping patterns as illustrated in Algorithm 4.

5 Evaluation and Performance Results

In this section, our proposed DQR is analyzed under different conditions. Various simulation scenarios have been run to prove the merit of DQR.

5.1 Experimental Setup

A WMSN for a ward in a field hospital has been adopted, as shown in Fig. 1. The SNs have been distributed randomly over an area of 50 m × 10 m. A total number of 64 SNs has been used where one of them acts as a sink, which represents the maximum number of SNs according to IEEE 802.15.6 [12]. The traffic is randomly generated using an exponential distribution density function.

Algorithm 4: Secure Trust Evaluation

Input: Observations & beta shape parameters
Output: Trust value
initialization;
while $TRUE$ **do**
  if $b_{t-1} \leq 0$ $\&\&$ $d_{t-1} > 0$ **then**
    $\alpha_t = \lambda(\alpha_{t-1} + b_{t-1}) + s_t$;
    $\beta_t = \lambda(\beta_{t-1} + d_{t-1}) + u_t$;
    $b_t = \alpha_t - \alpha_{t-1}$;
    $d_t = \beta_t - \beta_{t-1}$;
  else
    $\alpha_t = \lambda.\alpha_{t-1} + s_t$;
    $\beta_t = \lambda.\beta_{t-1} + u_t$;
    $b_t = \alpha_t - \alpha_{t-1}$;
    $d_t = \beta_t - \beta_{t-1}$;
  end
  if $\alpha_t \leq 0$ **then**
    $Rep_t^{ij} = 0$;
  else
    $Rep_t^{ij} = \frac{\alpha_t}{\alpha_t + \beta_t}$;
  end
  if $T_{t-1}^{ij} \geq thr_1$ $\&\&$ $Rep_t^{ij} < thr_1$ **then**
    if $malicious > 0$ **then**
      $cycle = t - malicious$;
      $malicious = 0$;
    else
      $malicious = t$;
    end
  end
  if $cycle > 0$ $\&\&$ $Trust(t-1) < thr_2$ **then**
    $ShRep_t^{ij} = mean(T_{t-cycle:t}^{ij})$;
    $T_t^{ij} = min(ShRep_t^{ij}, Rep_t^{ij})$;
  else
    $T_t^{ij} = Rep_t^{ij}$;
    $cycle = 0$;
  end
end

DQR is benchmarked with QRT [11] routing protocol, which has been designed to handle non-cooperative and misbehaving SNs in WMSN. It has been proposed as a trust-based extension to RL_QRP [14], an RL-based routing protocol proposed to fit WMSN. QRT has been chosen as a benchmark because it is the only routing protocol proposed to deal with dropping attacks in WMSN. To ensure fair comparisons, we adopted the reported parameters setting of QRT as shown in Table 1. The experiments have been run using a discrete event simulator based on Simpy [18]. The simulation time is set to $200s$ where the first $50s$ represents the learning time. The exploration-exploitation rate is controlled by ε-greedy strategy and set to 10% as in QRT. Each experiment has been run 30 times to ensure the Gaussian distribution. The results are then averaged out and reported with one standard deviation.

5.2 Delivery Reliability Analysis

In these experiments, the delivery performance is evaluated under different network conditions ranging from normal operation to under complicated attacks.

The packet delivery ratio and hop counts are considered to compare the optimality of the routing decisions made by both protocols.

Table 1. Simulation parameters

Parameter	Value
Application	Poisson random traffic
Traffic rate μ	1, 2, 4, 8
Radio range	5 m
Propagation loss model	Range propagation loss
Number of SNs	64
Time unit	1 s
Simulation time	200 s
Learning period	50 s
Learning rate η	0.5
Discount factor γ	0.5
$\varepsilon-$greedy	0.1
The average weight ω	0.5
Residual energy threshold ϑ	0.7

The first experiment studies the performance under normal operation with variable traffic rates. Some SNs generate a low traffic rate of around $1p/s$, such as heart rate SNs [19]. Thus, four traffic rates have been chosen for simulation, starting at $1p/s$ and doubling it each time. No malicious SNs are considered in this experiment. Benign nodes may drop randomly 1% of the traffic. Figure 3a and 3d show the delivery ratio, and the hop counts for both protocols under normal operation, respectively. DQR show superior data delivery performance with optimum routing decision. QRT shows high variability in terms of delivery ratio and hop count, which indicates that QRT does not converge to the optimum action-value(s) all the time. Moreover, it finds difficulty working under low traffic rates.

In the second experiment, blackhole and selective forwarding attacks are launched during the simulation to study the robustness of both protocols. The blackhole attack is a dropping attack where malicious nodes drop all received traffic instead of relaying it. In the selective forwarding attack, the malicious nodes selectively choose some sources to drop their traffic. Both attacks may disrupt the network operation. Therefore, nodes should always choose the most reliable path to destinations. The performance has been evaluated for a variable number of malicious nodes, starting from 1 malicious nodes and up to 50% of the total number of SNs. Figure 3b, 3c, 3e and 3g shows the delivery ratio and the hop counts under blackhole and selective forwarding attacks, respectively. Across all scenarios, DQR chooses the most optimal reliable paths, as illustrated in the

hop counts results. When the number of malicious nodes increases, DQR avoids malicious paths and tends to choose longer but reliable paths. On the other hand, QRT is not able to detect malicious paths, as shown in the decreasing hop counts when introducing more malicious nodes. This means that packets end up in malicious nodes, which explains the low delivery ratio.

In the third scenario, sinkhole, which is a route poisoning attack, is launched to study the impact of receiving dishonest updates from other agents on routing decisions. Different levels of poisoning are evaluated starting by increasing the updates by 25% and doubling it up to 100%, where the agents send the value zero, which is the highest Q value in DQR. The delivery and hop counts ratios are illustrated in Fig. 4a–Fig. 4f. The results show that DQR is robust under different poisoning levels and can achieve a high delivery ratio. It is worth noting that in the worst-case scenario when malicious SNs advertise zeros, DQR takes slightly longer paths as the received false updates influence not only the SN itself but also its neighbors. However, this behaviour does not affect the delivery ratio.

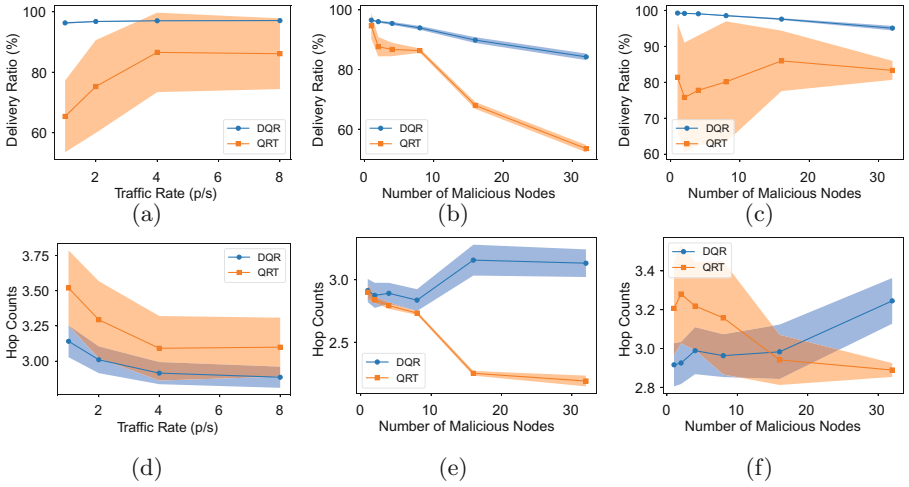


Fig. 3. Delivery and hop counts ratios under different conditions

5.3 Convergence

Q-learning is proved to converge to the optimum action-value(s) [20], as is double Q-learning [5]. However, convergence time is a crucial factor. A longer time to converge implies risking more packets to lose and consuming extra resources. In this experiment, the convergence time is evaluated in two scenarios, at the beginning of the simulation and when patients change their locations. Figure 5a demonstrates the convergence time at the beginning of the simulation, where SNs have no information about the environment and need to explore in order to converge. DQR converges faster than QRT thanks to its asynchronous updating algorithm. It took less than 50% time to converge compared to QRT. It is worth

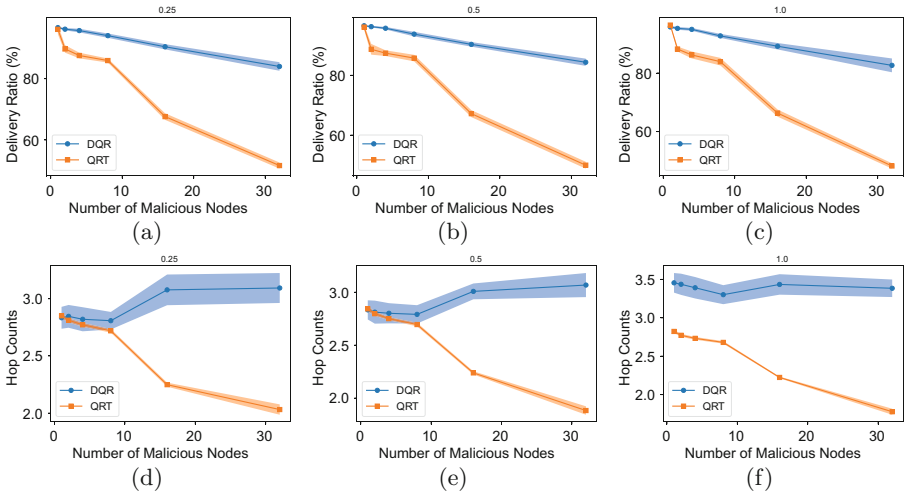


Fig. 4. Delivery and hop counts ratios under sinkhole attack

noting that QRT performs well at the early start because it is provided by positional information, while DQR works without any prior knowledge. In the second scenario, patient mobility is considered within the hospital ward. The patient could have up to three SNs. Thus the simulation is run for 1, 2 or 3 randomly chosen SNs at a time. Two movements have been considered at times $100s$ and $150s$. The simulation has been run for a hostile environment where 50% of the nodes are launching blackhole attacks. Figure 5b shows mobility results for only three SNs due to space constraints. The results show a fast convergence without any noticeable performance degradation for DQR protocol, which proves the robustness of our methods. On the other hand, QRT suffers from difficulty in re-converging, especially after the second movement.

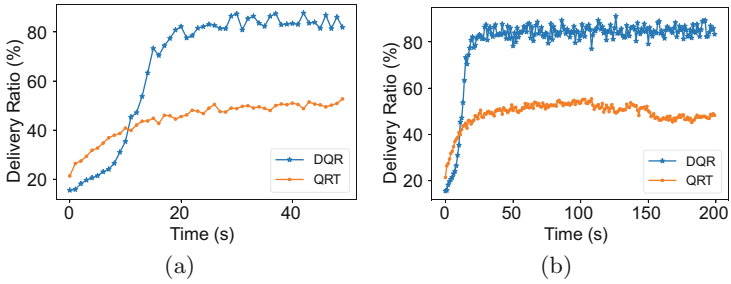
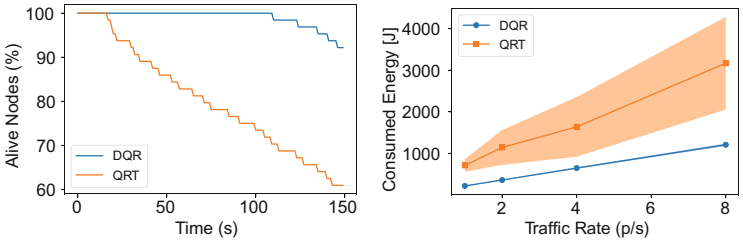


Fig. 5. The average convergence time

5.4 Energy Efficiency

The energy efficiency has been evaluated in two experiments. In the first, the network lifetime has been compared between both protocols. The second scenario shows the average consumed energy by a node for different traffic rates. Network lifetime could be defined as the running time until the first node dies [7]. Both simulation scenarios have been carried out under normal operation without introducing any attack. Figure 6a shows the percentage of alive nodes during the simulation. QRT has a very short lifetime compared to DQR. The first node dies after around 16s on average. This deficiency could be attributed to two reasons. QRT does not take any energy-related factor into account to choose the optimal path, and most importantly, the excessive information exchanging increases the RF activities significantly, which is responsible for 80% of the consumed energy. On the other hand, DQR shows superior performance because of its resource-conservative design, which is clearly reflected in consuming less energy for all traffic rates, as obviously seen in Fig. 6b.



(a) (b)

Fig. 6. Energy efficiency results

5.5 Computational Overhead

In this experiment, the processing and memory overheads of both protocols are evaluated. The experiment was carried out on an Intel Core i9-10885H at 2.4 GHz and 32 GB RAM. The computational overhead has been evaluated for traffic rate $\mu = 4p/s$ as QRT has converging difficulties for lower rates. No attacks were launched during the simulation. The simulation was repeated 30 times, and then the mean with one standard deviation was reported.

The average processing time of both protocols is illustrated in Fig. 7a. What can be clearly seen in this result is the minimum processing overhead of DQR. It saves around 50% processing overhead compared to QRT. Moreover, unlike QRT, DQR has minimum variability. This proves that our novel RL model is resource-efficient. Moreover, the low variability of DQR indicates that DQR is always able to converge swiftly without any difficulties, proving its robustness.

Memory consumption is another crucial factor for constrained devices, such as SNs. Figure 7b depicts the average consumed memory of both protocols. During the simulation, the memory allocations were traced using a memory allocation module called tracemalloc [21]. The results show that DQR is able to save up to 80% of QRT consumed memory. Moreover, unlike QRT, DQR shows almost no variability, which indicates its robustness.

The results of this experiment show that DQR has a minimum footprint in terms of processing time and consumed memory. This lightweight computational overhead could be attributed to its resource-efficient design represented in synchronous and asynchronous Q tables updating algorithms. Furthermore, this novel design is able to converge swiftly with minimum variability allowing the packets to reach their destinations efficiently.

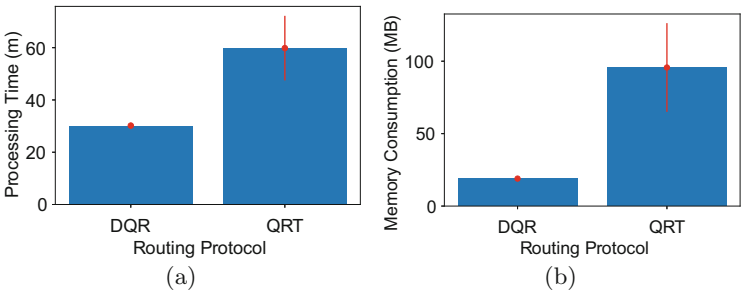


Fig. 7. The average processing time and memory consumption

6 Conclusion and Future Work

The resource scarcity and the sensitive applications have brought enormous challenges to WMSN routing protocols. The existing routing protocols for WSN cannot be directly adopted for WMSN due to overlooking some imperative requirements. In this paper, a lightweight, reliable and energy-efficient routing protocol for WMSN has been proposed. DQR is a double Q-learning routing protocol that uses a novel RL model. It uses two updating methods combined with trust management and energy models to ensure lightweight, reliable and resource-efficient data delivery. The experimental results show superior performance with minimal resource footprint. The performance of DQR will be further optimized by tuning the used hyper-parameters. Additionally, more experiments will be carried out to ensure robustness under further complicated dropping attacks.

References

1. Barker, A., Swany, M.: Distributed cooperative reinforcement learning for wireless sensor network routing. In: 2022 IEEE Wireless Communications and Networking Conference (WCNC), pp. 2565–2570 (2022)
2. Keerthika, A., Berlin Hency, V.: Reinforcement-learning based energy efficient optimized routing protocol for WSN. Peer-to-Peer Network. Appl. **15**, 1685–1704 (2022)
3. Mammeri, Z.: Reinforcement learning based routing in networks: review and classification of approaches. IEEE Access **7**, 55916–55950 (2019)
4. Künzel, G., Indrusiak, L.S., Pereira, C.E.: Latency and lifetime enhancements in industrial wireless sensor networks: a Q-learning approach for graph routing. IEEE Trans. Ind. Inform. **16**, 5617–5625 (2020)
5. Hasselt, H.: Double Q-learning. In: Advances in Neural Information Processing Systems, vol. 23 (2010)
6. Yuan, F., Wu, J., Zhou, H., Liu, L.: A double Q-learning routing in delay tolerant networks in ICC 2019. In: 2019 IEEE International Conference on Communications (ICC), 1–6 (2019)
7. Guo, W., Yan, C., Lu, T.: Optimizing the lifetime of wireless sensor networks via reinforcement-learning-based routing. Int. J. Distrib. Sens. Netw. **15**, 1550147719833541 (2019)
8. Maivizhi, R., Yogesh, P.: Q-learning based routing for in-network aggregation in wireless sensor networks. Wireless Netw., 2231–2250 (2021)
9. Al-Rawi, H.A., Ng, M.A., Yau, K.-L.A.: Application of reinforcement learning to routing in distributed wireless networks: a review. Artif. Intell. Rev. **43**, 381–416 (2015)
10. Liu, G., Wang, X., Li, X., Hao, J., Feng, Z.: ESRQ: an efficient secure routing method in wireless sensor networks based on Q-learning. In: 2018 17th IEEE International Conference on Trust, Security And Privacy in Computing and Communications/12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE), pp. 149–155 (2018)
11. Naputta, Y., Usaha, W.: RL-based routing in biomedical mobile wireless sensor networks using trust and reputation. In: 2012 International Symposium on Wireless Communication Systems (ISWCS), 521–525 (2012)
12. IEEE. IEEE Standard for Local and metropolitan area networks - Part 15.6: Wireless Body Area Networks. IEEE Std 802.15.6-2012, 1–271, February 2012
13. Azdad, N., Elboukhari, M.: Wireless body area networks for healthcare: application trends and MAC technologies. Int. J. Bus. Data Commun. Network. (IJBDCN) **17**, 1–20 (2021)
14. Liang, X., Balasingham, I., Byun, S.-S.: A reinforcement learning based routing protocol with QoS support for biomedical sensor networks. In: 2008 First International Symposium on Applied Sciences on Biomedical and Communication Technologies, pp. 1–5 (2008)
15. Jiang, J., Zhu, X., Han, G., Guizani, M., Shu, L.: A dynamic trust evaluation and update mechanism based on C4. 5 decision tree in underwater wireless sensor networks. IEEE Trans. Veh. Technol. **69**, 9031–9040 (2020)
16. Krishnaswamy, V., Manvi, S.S.: Trusted node selection in clusters for underwater wireless acoustic sensor networks using fuzzy logic. Phys. Commun. **47**, 101388 (2021)

17. Hajar, M.S., Al-Kadri, M.O., Kalutarage, H.: LTMS: a lightweight trust management system for wireless medical sensor networks. In: 2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom), pp. 1783–1790 (2020)
18. Matloff, N.: Introduction to discrete-event simulation and the SimPy language. In: Davis, C.A. (ed.) Department of Computer Science. University of California at Davis (2008). Accessed 2 Aug
19. Islam, M.N., Yuce, M.R.: Review of medical implant communication system (MICS) band and network. Ict Express **2**, 188–194 (2016)
20. Melo, F.S.: Convergence of Q-learning: a simple proof. Institute of Systems and Robotics, Technical Report, 1–4 (2001)
21. TRACEMALLOC - Trace memory allocations - Python 3.10.2 documentation. https://docs.python.org/3/library/tracemalloc.html. Accessed 8 Feb 2022