Energy Optimisation Through Machine Learning in Wireless Networks

Adarsh Denga
University of Twente
P.O. Box 217, 7500AE Enschede
The Netherlands
a.a.denga@student.utwente.nl

ABSTRACT

The world around us is filled with internet of things (IoT) devices trying to communicate with each other, often in environments that are dynamic and unknown. Thus, there is a need for the behaviour of these devices to be altered such that the quality of service (QoS) of these devices is maximised. This research will focus on researching the multi-armed bandit to optimise the energy usage of the nodes in a wireless network in order to maximise the longevity of individual nodes. The implementation of this project will involve gathering data from OMNeT++ and running the multi-armed bandit to optimise behaviour of a node and derive results from.

Keywords

Wireless Sensor Networks, Machine Learning, Deep Learning, Deep Neural Networks, Multi-Armed Bandits, Federated Machine Learning

1. INTRODUCTION

Since the dawn of ubiquitous computing we have lived in a world where the number of electronic devices around us has been increasing at a rapid rate. A 2020 study shows that there are around 12 billion active IoT devices in the world, and this number is estimated to grow to around 31 billion by 2025 [5]. Even within our own homes, there are various devices connecting with one another over the Internet. Quite often, these devices are placed and expected to function in environments that are dynamic and unknown.

Some nodes are affected more than others by the unknown and dynamic nature of the environment around us, due to additional limitations in the form of limited processing and computational power, and most importantly, limited battery lives. This makes the problem of energy optimisation for such nodes especially relevant, as limited battery power is a major hindrance to the potential of wireless sensor networks (WSN).

In this research I will investigate the multi-armed bandit machine learning algorithm to arrive at an optimal solution for the problem of energy optimisation for one node. Data for this purpose will be gathered from OMNeT++ and then used to train the multi-armed bandit to attempt the problem of energy optimisation.

28^{th} *Twente Student Conference on IT* Febr. 2^{nd}, 2018, Enschede, The Netherlands.

2. RELATED WORK

Due to the wide variety of applications for WSNs and wireless networks in general, there is a strong motivation for research into machine learning algorithms for the optimisation of such networks.

Primitive (non-intelligent) attempts at optimisation such as the ones in [1, 2, 6] attempt to optimise energy usage in WSNs, but all have their drawbacks. For instance, the LEACH hierarchical clustering protocol [2] fails to choose cluster-heads intelligently, resulting in sub-optimal power usage. Wang et al. in [6] propose a novel energy-aware hierarchical cluster-based routing protocol for WSNs where cluster heads are obtained as a function of the residual energy in nodes as well as their proximity to other nodes, but it is speculated that this approach would not be optimal in larger networks with many nodes [1].

The difficulty of the resource allocation problem makes machine learning an attractive approach for optimisation. Early approaches to optimisation of node behaviour using intelligence involved the use of deep learning (DL) tools in order to produce optimal resource allocation solutions. Both supervised and unsupervised learning methods have been investigated for this purpose.

The supervised method studied in [11] tries to use a deep neural network (DNN) in order to study a local network solution developed by a weighted minimum-mean-square-error (WMMSE) algorithm. This approach is strong, but its drawbacks lie in the fact that it requires a large number of training labels for the DNN to "learn to optimise". Unsupervised methods such as those in [3, 7, 8] are good at arriving at solutions that are far less computationally intensive than the WMMSE algorithm.

In general, most DL approaches to intelligent optimisation in WSNs require global knowledge of the network and the channel state information (CSI) between nodes for a DNN to develop a solution. Gathering global data about all the nodes of a network is not practical due to the limited cooperation between nodes. Therefore, a centralised approach is not realistic.

Distributed approaches the the problem of optimisation in wireless networks do exist. For instance, the distributed method in [8] has a DNN that is trained in a centralised manner and uses zeroes for unknown CSI inputs while testing, which degrades its performance greatly. Another such distributed approach [3] distributes a dedicated DNN to each node after training, but fails to optimise the CSI estimation and the process of exchanging data. Thus, the problem of finding a distributed DL approach to network energy optimisation still remains open.

3. PROBLEM STATEMENT

Although there has been significant research into the optimisation of the behaviour of nodes in wireless networks in order to maximise the QoS, the problem of energy optimisation in wireless networks still remains open.

The optimisation of energy can be done at multiple levels, such as the physical layer (e.g. frequency/amplitude modulation), MAC layer (e.g. medium access protocols), network layer (e.g. energy-efficient routing), the application layer (e.g. data aggregation) [10]. Research will be conducted to determine which of the above approaches can be used in order to prolong network lifetime as well as maintain low computational complexity at the nodes.

The optimisation in this research will focus on developing a network solution for a single node by using the CSI it can gather from its neighbours, as compared to a centralised approach with a single node developing a solution for the entire network. In a centralised approach, all the nodes in a network would be able to send their data to a central hub which could be equipped with higher processing power in order to develop a global network solution for the purpose of optimisation. However, due to limitations in connectivity range, limited capacities of backhaul links and so on, gathering and transmitting perfect CSI of all the nodes in a network is unrealistic.

Arriving at a machine learning solution for energy optimisation for the nodes in a wireless network presents two main issues. These nodes have low computational power, which isn't suited for machine learning applications. Secondly, they have limited battery power.

By researching different flavours of machine learning algorithms, I intend to arrive at a solution that is an optimal policy for the management of energy for one node in a network. Furthermore, this machine learning algorithm must be computationally inexpensive as well, in order to be able to reduce network energy usage as a whole. Such a solution could be expanded upon in order to develop a solution for the entire network. As such, though the research has deviated from its initial goal of arriving at a distributed solution for a network, the algorithm investigated will be one that could be used in a distributed setting as well.

The optimisation focuses on making a node more power efficient. From here onward, any mentions of optimisation will imply the reduction of the Joules of energy used per acknowledged byte of data, given by the metric

$$Joules/Byte(Acknowledged)$$

3.1 Research Question

The greater research question can be stated as follows:

- RQ: Can a distributed machine learning algorithm change the behaviour of the nodes (as in the optimisation strategies given above) in order to prolong the lifetime of a network of wireless nodes?

This question can further be split into the following subquestions, which need to be answered in order to present the global solution:

- RQ1: Which machine learning algorithm(s) can work best in a distributed manner?
- RQ2: Are they computationally inexpensive in the context of the limited processing and battery power available?
- RQ3: Can they present an optimal solution that when implemented will reduce the energy usage for both the individual node and the network as a whole?

4. SIMULATOR

The simulator that will be used for the purpose of network set up, data extraction, testing and so on will be the OMNeT++, which is a C++ based simulation library and framework. Since the technology of choice involves a network of nodes that communicate over Wi-Fi, INET is used. INET is an open-source OMNeT++ model that is used for wireless and mobile networks.

5. NETWORK SETUP

Figure 1 shows the general network setup with the nodes, the other modules such as the configurator, medium and visualizer. The sections below will explain in detail the simulation technologies used to produce the final simulation.

5.1 Nodes

The nodes in the simulator are spread out in an area that is 650 metres by 500 metres. There are 5 nodes in the space. These nodes are of the StandardHost type. The INET framework has variations of the StandardHost type, and for the purpose of the simulation, a specific variation called the WirelessHost is used in order to model wireless communication. In practice, this type is further specified into the AdhocHost type, which is a wireless host that contains routing, mobility and energy components. It supports IPv4, UDP, and can have UDP applications installed. For the purpose of the research, UdpBasicApp and UdpSink applications are installed on different nodes in the network.

5.2 Address Assignment

IP addresses are assigned to each of the nodes using INET's IPv4NetworkConfigurator module. In order to communicate with one another, the nodes also need to know a link layer address, such as a MAC address. For the discovery of the MAC addresses of other nodes, the model uses per-host GlobalArp, which is an OMNeT++ module to discover global address resolution without having to exchange packets.

5.3 Ad-Hoc Routing

Instead of adding static routing tables to determine connections between nodes, AODV is used to dynamically maintain routes while they are needed. Although the nodes in this simulation will remain static, AODV is used as it is more representative of a real life network of nodes, some of which may be mobile. Packet forwarding is also enabled, so that packets can be sent to the receiver even if a direct route is not available.

5.4 Acknowledgements

Acknowledgements are set up in the network so as to confirm the reception of packets. On the receiver side, when the MAC correctly receives a data frame that is addressed to it, it will respond with a CsmaAck frame that is sent back to the transmitter node. The MAC module used is the INET CsmaCaMac module. In this module, a new packet is only transmitted when the currently transmitted packet is acknowledged.

5.5 Traffic Modelling

The transmitter nodes generate UDP packets that are received by the other nodes. The transmitter nodes are con-

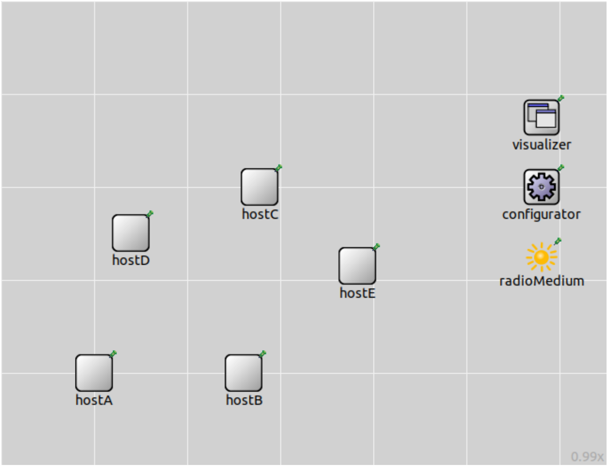

Figure 1: Network setup

figured to contain an UdpBasicApp module which generates 1000 byte UDP messages. These messages are generated at random intervals with an exponential distribution, where the mean is 12ms. As such, the UdpBasicApp will generate 800kbps UDP traffic (excluding protocol overhead). The receiver nodes contain UdpSink applications that discard received packets.

5.6 Physical Layer

All wireless simulations that use INET need a radio medium module. The radio medium module represents the physical space that the nodes share. It is responsible for taking physical phenomena such as signal propagation, attenuation and interference into account.

The radio medium module that is used for the purpose of this simulation is called the ApskScalarRadio, which has an APSK (amplitude and phase-shifting keying) modulation scheme. This specific module allows for the medium to be realistic, by allowing the simulation of signal attenuation, background noise, and interference. Furthermore, it also allows for the modification of node behaviour through the specification of parameters such as carrier frequency, signal bandwidth and transmission power.

6. DATA EXTRACTION

Data extraction from the simulation is done by getting the raw packet data from the simulator after the simulation has completed and then manipulating it using a Python parser to better fit the machine learning algorithm.

The complete process of extraction is as follows. The packet data is copied over from the simulator into a text file, and the Python parser is given the final values of energy usage for each node. Then, the parser combs through each line of the file and pre-formats each line to make the data easier to work with. It then splits up the data by sender and calculates the total amount of bytes each sender has sent. It then uses this statistic to find the approximate energy usage per byte. Finally, it assigns the energy usage per packet by simply multiplying the energy used per byte into the size of the packet in bytes. One drawback of gathering all the data after the simulation is over is that energy readings cannot be taken live, and therefore an estimation must be made, rather than gathering data in real time.

Given that the extracted data will contain the energy usage per packet, and that the number of packets received is known, it will be a good fit for the machine learning algorithm which aims to optimise the final metric of energy usage per quantity of data sent (Joules/Byte).

7. MACHINE LEARNING ALGORITHM

The machine learning algorithm chosen for the implementation is the multi-armed bandit. The choice of this specific algorithm over other, more complex ones, such as types of reinforcement learning, is due to the performance required in the context of the devices themselves. The algorithm is aimed to be run on edge devices which have limited processing and battery power, and given the low amount of calculations and speed of the algorithm itself, the multi-armed bandit is an attractive option for the such devices.

7.1 Federated Learning

Federated learning is a machine learning approach where the task of developing a machine learning algorithm is split across multiple decentralized edge devices [12]. In the context of this research, this would involve each of the nodes gathering their own local data they observe and then developing a part of the greater algorithm using their local chunk of data. As stated previously, a major hindrance to the potential of wireless networks is limited processing power. This makes the task of developing federated intelligence especially tricky, as each node must develop an algorithm that is computationally inexpensive.

7.2 Multi-Armed Bandits

The multi-armed bandit is a problem in which a fixed set of resources must be allocated between competing alternate choices in a way that maximises the expected gain [9]. This algorithm roughly matches the goal of the optimisation problem in this research. Given the usage of Wi-Fi as the technology of choice for the purpose of this research, the optimisation of such a network can be done in multiple ways. For the purpose of this research however, the multi-armed bandit will try to optimise the network by changing the transmit power of the nodes. Each node transmits by default at 1.4mW, but the bandit can switch to a lower power mode of 0.7mW to check if that produces more optimal results.

In short, the multi-armed bandit algorithm works as follows. There is a feature that needs to be optimised, which in this case is the Joules of energy used per byte of data acknowledged. There is a data set consisting of packet information and the energy that is used to send the packet. At the start, the algorithm does not know the impact that changing the transmit power will have. In order to both learn and optimise, the algorithm will have to split its time between exploring options and their consequences, as well as exploiting the data that it gathers through learning.

The key in optimisation through this algorithm is striking a balance in the time the algorithm devotes to exploration and exploitation. The quantity that describes this split is denoted by ϵ. If $\epsilon = 10\%$, then the algorithm will explore for 10% of the time and exploit for 90% of the time. The goal of the algorithm is to reduce the metric of the optimisation, which is the Joules per byte acknowledged for both power levels and overall. As a whole, this algorithm is called the ϵ-Greedy algorithm. The mathematical notation below shows a basic overview of the ϵ-Greedy algorithm, with the probabilities of each path the algorithm can take.

$$p(explore) = \epsilon$$

$$p(exploit) = 1 - \epsilon$$

$$p(1.4mW/explore) = 0.5$$

$$p(0.7mW/explore) = 0.5$$

7.3 Network Setup and Machine Learning Algorithm

Out of the five nodes, as shown in Figure 1, host A is configured as the transmitter, and host E is configured as the receiver. It is considered that each node can have two power levels, normal and low. At normal, nodes transmit at 1.4mW, and at low, nodes transmit at 0.7mW.

In a real life situation, nodes at different power levels would be influenced by each other, leading to differing overall power efficiencies. For the sake of the research, the problem is simplified as follows. The problem is considered only for the transmitting node. Therefore, two sets of data have been recorded where A transmits at both 1.4mW and 0.7mW, and all the other nodes transmitting at 1.4mW.

As such, the multi armed bandit is equipped with two 'arms' - it can choose to switch between the two power nodes to see how the final power efficiency of the transmitter node is affected. An important thing to consider here is the reliability of the packet transfer as well. By the algorithm, the efficiency of a particular power mode is only counted in terms the Joules of energy used per byte acknowledged, not sent. Therefore, by default, the algorithm will also pick the more reliable of the energy modes.

The simulation here is performed offline, which means the two data sets at the different power levels are taken to be a representation of the simulation's actual run. The reason for this is that an OMNeT++ simulation is started with a specific network configuration (stating parameters like transmit power for each node, etc.) and changing these midway though a simulation every time the algorithm makes a decision is a tedious process.

7.4 Multi-Armed Bandit Implementation

In reality, the multi-armed bandit implemented for this research can be considered a two-armed bandit, as it works to choose between two power levels, 1.4mW and 0.7mW. The bandit is implemented on Jupyter Notebook.

7.4.1 Data

The data for the bandit is from two different csv files, for the high and low power levels. For the sake of the algorithm implemented, the data used to optimise on a single node is the timestamp of a packet, the source node, the type of data (UDP data / acknowledgements / AODV packets), the size of the packet, and the energy used for the transmission of that packet.

	1.4mW	0.7mW	Bandit
Bytes Acknowledged (B)	1063	1063	2126
Energy Used ($\times 10^{-3} J$)	7.82	7.56	7.69

Table 1: Metrics tracked by the Multi-Armed Bandit

7.4.2 Metric Tracking

As the program runs, it keeps track of three different metrics. Two of these metrics are the power efficiencies of the each energy level. They track the total energy used for each power level as well as the number of acknowledged bytes of data for that energy level. The third is an overall metric which keeps track of the real time power efficiency as the algorithm 'runs' through the simulation, which will determine finally if the algorithm can actually produce sufficient optimisation. Table 1 shows the metrics tracked by the algorithm as well as the values soon after the simulation has started.

7.4.3 Explore vs Exploit

The bandit tries to simulate a 'run' of the network by going through the chronological order of the packets. At every given step, the bandit has the option to either explore or exploit. Whether it chooses to explore or exploit depends on a random variable which will generate either the number 0 (to explore) or the number 1 (to exploit) based on the probability given in ϵ. For instance, if the value of ϵ is 0.10, then the algorithm will explore 10% of the time and exploit based on exploration data for 90% of the time.

7.4.4 Explore

When the bandit has to explore, there is an equal probability that either power mode will be chosen. This is determined by a simple random number generator. When a power level is determined, the simulation will 'run' at that power level. This means that a packet will be read from the file corresponding to that power level and be used in the calculation of the power metrics for each power level, as well as the overall metric of the algorithm itself.

7.4.5 Exploit

When the bandit must exploit, it will refer to the above mentioned metrics for each power level (Section 8.4.2) and choose to run the simulation at the power level that has the better power efficiency out of the two (lower Joules/Byte). Then, similar to the exploration strategy, it will add the power usage and acknowledged bytes to the metrics for the algorithm to learn from.

The 'run' of the simulation ends when there are no more packets to be sent from either file. The simulation does not come to a halt, but rather ceases after a time limit.

8. RESULTS

8.1 Implementation Results

As shown in Figure 2 there are four graphs. The x axis represents time in seconds. The simulation begins at 0 seconds and ends at 10 seconds. The y axis is the power efficiency, given in Joules/Byte acknowledged. The blue line in each of the graphs represents the running power efficiency when the transmitter is transmitting at 1.4mW, the high power level. The orange line in each of the graphs represents the running power efficiency when the transmitter is transmitting at 0.7mW, the low power level. The green line represents the actual power efficiency achieved by the multi-armed bandit. Given the way the CsmaCaMac works, where a new packet is only sent when the old one is acknowledged, all the packets are received. However, it may take multiple re-transmissions for a packet to finally be received. As such, the algorithm will also tend to choose the power level with the lower amount of re-transmissions, as energy is used for each re-transmission without acknowledgement.

All four of these graphs follow the expected trend. The power efficiency varies to a high degree initially, as the bandit spends time to explore the network and learn the efficiencies of the power levels. When it has finished its initial exploration, it continues, exploiting when it is chosen to. As expected, after the initial exploration, the running power efficiency of the algorithm reduces and converges near the power efficiency of the low power transmissions.

Also seen in the graph is that the higher the value of ϵ, the more often the efficiency of the high power transmission varies. This is due to the fact that a higher value of ϵ implies more chances of exploring during any given step. However in between these exploration steps, the efficiency of the higher power level remains unchanged, as it is abandoned in favour of the more power efficient low power mode.

8.2 Research Questions

8.2.1 RQ1

Given the need for fast optimisation and the limited information about the network, the multi-armed bandit class of machine learning algorithms are a good fit for such optimisation problems. The multi-armed bandit is designed to make quick decisions based on limited information, which could even work on something 'live' like a simulation or node behaviour in an actual network. So, given that the bandit can learn as it gathers more data, it makes it a natural fit for a network optimisation problem where data is gathered as time passes.

8.2.2 RQ2

Given that the algorithm was run on a computer, it may be hard to determine if edge devices in networks can run the developed algorithm efficiently enough to optimise communication. However, the algorithm is indeed very light. At each step, it only needs to decide whether to explore or exploit, and then which power level to run the simulation at. The metrics must also be updated to give the algorithm data to learn from.

In fact, the algorithm that has been implemented carries needless weight as a result of it being offline. In a real life scenario, a lot of the steps performed can be skipped, as packet data from the network can be configured to readily have the information needed to make an optimisation.

8.2.3 RQ3

As of yet, the the bandit implemented can only produce an optimal solution for a single node. However, this bandit is able to make swift decisions based on network conditions and can regulate the behaviour of a node in order to greatly reduce its power consumption for every byte of data that it successfully sends.

Given that the behaviour of the algorithm is random (simply due to the nature of the algorithm itself) each run of the algorithm will yield a different amount of power optimisation. Therefore, it is challenging to accurately state a figure as to how much power it saves per acknowledged byte of data.

8.2.4 RQ

The current solution, though isolated and only for a particular node, is distributed - each node gathers its local channel state information as well as packet data and then aims to optimise its own energy consumption. The solution is also very lightweight and can be used in a live environment from a standstill, with the algorithm knowing little about the network.

Therefore, this algorithm is a good fit for network energy optimisation in a distributed manner. It is able to learn from the data how reliable each transmit power is and can make decisions that both further help it learn as well as simultaneously optimise the network.

8.3 Future Work

Some future additions to this research can be the development of a federated learning system based on the multi-armed bandit implementation. Each node trains its chunk of the algorithm based on data it observes, which contributes to a larger, global algorithm for the entire network.

The network considered in this research is quite simple. So, it would be interesting to scale up the number of nodes, change the number of transmitters and receivers, and also add mobility in the network to see if the algorithm can still optimise communication.

Given that OMNeT++ has the ability to interface with Python, the algorithm could also be implemented to work 'live' with the simulation, optimising and altering the behaviour of the actual nodes in the simulation as it runs.

Another interesting experiment can be the implementation of the ϵ-Decay algorithm [4], which reduces the value of ϵ as time passes. The reason for this is that the algorithm needs to learn less and less as time passes once it has found an optimal route. However, it could be argued that this can actually be detrimental to the performance of the algorithm in this specific application. This is because network conditions are not fixed, given that nodes can move in and out of range, and a host of other environmental factors. Therefore, the algorithm may not be able to rely on a previously learned solution as it may not apply to changing network conditions.

9. CONCLUSION

The problem of energy optimisation is especially relevant in the current time, with WSNs and wireless devices proliferating more and more areas of our lives. Since limited battery and computing power reduce the potential of these networks, there is a great need to study methods of optimisation that can be run in a distributed manner to prolong the lifespan of these networks.

This research has taken a look at one particular algorithm, the multi-armed bandit, and has implemented it on network data gathered from a simulator. The implemented algorithm is efficient and can make swift decisions with very little data to begin with. The decisions it makes positively influence the network efficiency both in terms of the energy used as well as the reliability of packet transfer. This makes such an algorithm a very attractive option for the field, especially in the context of the limited processing power as well as the dynamic and unknown nature of networks in real life.

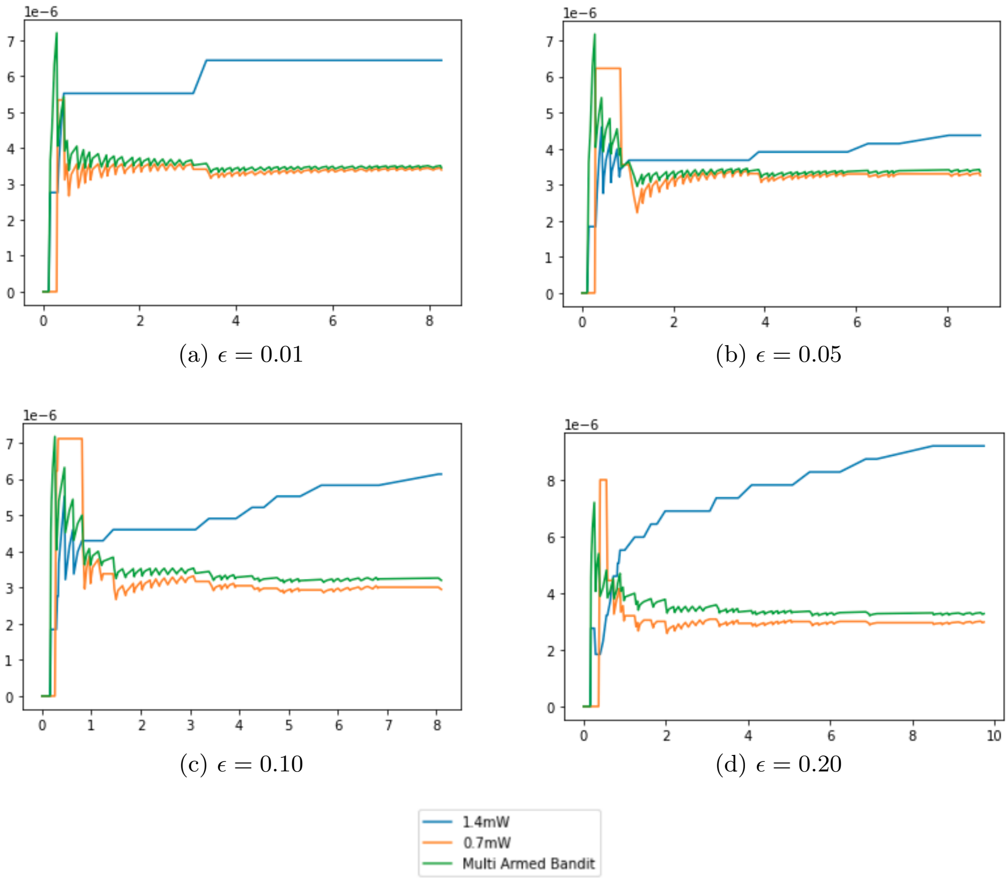

Figure 2: Multi-armed bandit performance for different values of Epsilon

10. REFERENCES

[1] A. P. Abidoye and B. Kabaso. Energy-efficient hierarchical routing in wireless sensor networks based on fog computing. *Eurasip Journal on Wireless Communications and Networking*, 2021(1), 2021.

[2] O. ABOLADE, A. OKANDEJI, A. OYEDEJI, M. OSIFEKO, F. ONAIFO, and P. ALAO. Dynamic advanced node selection for wireless sensor networks for improved network life and packet transmission. *Scientific African*, 12, 2021.

[3] P. de Kerret, D. Gesbert, and M. Filippone. Team deep neural networks for interference channels. In *2018 IEEE International Conference on Communications Workshops (ICC Workshops)*, pages 1–6, 2018.

[4] C. Hubbs. Multi-armed bandits and reinforcement learning. 2019.

[5] IoT-Analytics. Internet of things (iot) and non-iot active device connections worldwide from 2010 to 2025 (in billions), 2020. Online: `https://www.statista.com/statistics/1101442/iot-number-of-connected-devices-worldwide/#:~:text=The%20total%20installed%20base%20of,that%20are%20expected%20in%202021.`

[6] W. Ke, O. Yangrui, J. Hong, Z. Heli, and L. Xi. Energy aware hierarchical cluster-based routing protocol for wsns. *The Journal of China Universities of Posts and Telecommunications*, 23(4):46–52, 2016.

[7] W. Lee. Resource allocation for multi-channel underlay cognitive radio network based on deep neural network. *IEEE Communications Letters*, PP:1–1, 07 2018.

[8] W. Lee, M. Kim, and D.-H. Cho. Deep power control: Transmit power control scheme based on convolutional neural network. *IEEE Communications Letters*, 22(6):1276–1279, 2018.

[9] ritvikmath. Multi-armed bandit : Data science concepts.

[10] H. Sharma, D. V. Sachan, and S. Akhtar. Performance analysis of modulation schemes for energy efficient wireless sensor networks. 11 2019.

[11] H. Sun, X. Chen, Q. Shi, M. Hong, X. Fu, and N. D. Sidiropoulos. Learning to optimize: Training deep neural networks for interference management. *IEEE Transactions on Signal Processing*, 66(20):5438–5453, 2018.

[12] Wikipedia contributors. Federated learning — Wikipedia, the free encyclopedia, 2021. [Online; accessed 13-June-2021].