博士論文 (要約)

Reinforcement Learning-Based Optimization in Energy Harvesting Wireless Sensor Nodes

(強化学習を用いた環境発電駆動センサーノードの最適化に関する研究)

シュレスタマリ　サソット

Shresthamali Shaswot

The University of Tokyo

December 2020

Abstract

Energy Harvesting Wireless Sensor Nodes (EHWSNs) constitute a major proportion of the rapidly growing number of connected devices in the Internet of Things (IoT). Given this scale of growth, autonomous operation of EHWSNs is now more of a requirement rather than a preferred feature. Since they harvest energy from their working environment, they do not require periodic energy replenishment and could therefore theoretically operate forever. With intelligent and adaptive policies, it is possible to extract maximum utility and uninterrupted operation, even when the working environment is complex and unpredictable.

Our work investigates Reinforcement Learning (RL) optimization methods for EHWSNs to learn to self-regulate its resources – especially with regards to its energy as this is the most critical resource in EHWSNs. This entails first - balancing the device energy consumption with the unpredictable energy supply, i.e., Energy Neutral Operation (ENO) and second - ensuring that the energy used has maximum utility for the user. Traditionally, analytic methods were used on account of their theoretical guarantees and low compute requirements. However, these methods have limited applications, are not adaptive, and require manual calibrations. More recently, RL methods have gained popularity mainly because of their adaptivity and general scope of application. However, RL methods suffer from high learning costs with respect to time, risk and computation which prevents them from being used in the real world. This is mainly due to non-optimal formulation of the RL problem (especially the reward function) and inefficient exploration. With this work, we tackle these limitations in three different orthogonal directions and contribute solutions to overcome them.

For our first contribution, we develop tabular RL solutions for a general EHWSN

that can generate near-optimal energy management policies and adapt to the working environment. We propose a general ENO reward function that abstracts the problem formulation from application-specific details. We demonstrate that even with the most basic RL methods and our novel reward function, we can expect adaptive behavior and superior policies.

Our second contribution proposes an efficient exploration strategy to minimize the learning costs that are required by Deep RL (DRL) to learn energy neutral policies. DRL replaces the look-up tables of tabular RL with more powerful neural network-based function approximators. In doing so, the problem formulation can accommodate continuous states and actions which extends it applicability. We propose a distributed learning solution that coordinates the many sensor nodes of a network to efficiently explore the large state space concurrently. Our method significantly minimizes learning risks and accelerates the learning process.

In the final part of our contribution, we introduce a general multi-objective RL (MORL) framework for EHWSN optimization. Modern EHWSNs are quite sophisticated and can uses its energy to for many different tasks (e.g., sensing, communication, and processing). Thus, it is necessary not only to ensure ENO but also to allocate energy between the different tasks within the constraints of a limited budget to maximize node utility. Current methods are unable to consolidate between the multiple objectives and tradeoff between them. Our MORL framework is more general and we show that provide solutions that can learn more intelligent policies with minimal learning costs compared to traditional RL methods. More importantly, with this framework, one can dynamically adjust the energy management policy to tradeoff during runtime between different performance objectives without additional training.

Thus, in our work we show that RL methods are very promising alternatives for optimization of EHWSNs. We show that with our novel problem formulation, they can learn to adapt to unknown environments for long-term uninterrupted operation. We also show that it is possible to minimize the learning costs by leveraging the distributed structures of nodes in the sensor network. Finally, we demonstrate that

with MORL, we can further use the RL approach to not only ensure ENO but also optimize the node utility by allowing runtime tradeoffs. With our contributions, EHWSNs are a step closer to autonomous and perpetual operation. As we inch closer to full autonomy, we can expect IoT to be more seamlessly integrated into our world and enrich our lives.

Acknowledgements

I would like to express my sincere gratitude to my supervisor Prof. Hiroshi Nakamura for mentoring me during my research. He provided me with the opportunity to come to Japan as a research student and supervised me during my Master's and PhD program. I have learned a lot from him and grown under his tutelage. I am also equally thankful to Prof. Masaaki Kondo, who has guided me at every step of my research tirelessly and patiently. His encouragement and advice have been instrumental in my research. Both Nakamura-sensei and Kondo-sensei have been a source of great inspiration and helped me establish the foundations of my research. Their strong ethics ,discipline, devotion to duty has inspired me to be a better person and a better researcher.

I have spent six years of my life as a member of the Nakamura-Kondo laboratory. In that time I have had the pleasure to meet a great many people who have been a part of the lab and they have all contributed in my research one way or the other. I am very grateful for their support. I would like to specially thank Ryuichi Sakamoto-sensei. Without his support, I doubt I could have got anything to work in the lab. I would also like to thank Yuan He-san for sharing his experience and valuable advice with me. I am also very grateful towards the office staff – Harumi Fujita-san, Kagari Seki-san and Mako Taki-san who have patiently helped me with the official procedures.

I thank the Department of Information Physics and Computing as well as the Graduate School of Information Science and Technology for accommodating me in their program and supporting me during my time as a student in the university. I would personally like to thank Mitsuko Yoshida-san from the department office who has always been extremely helpful with the paperwork and has gone beyond the call

of duty on my behalf. I would also like the staff at the International Student Support Room, especially Yamamoto-san. I am especially grateful towards Kaori Sato-san from Office of International Students, Graduate School of Information Science and Technology. She has been with me every step of the way starting from when I was just applying to the university as a prospective student. Her support and advice have been very invaluable during my experience as an international student in Japan.

I am grateful towards the university and its staff for providing me the opportunity and the platform for my research. Its excellent facilities have greatly aided me in my research and my success is limited only by my own capacity. The diligence of the university staff, from the cafeteria servers to the cleaning staff, have inspired me to always give my best and hold the highest standards of discipline and ethics.

During my time in Japan, I have had the pleasure of making a great many friends from around the globe and I would like to thank them for their love, support, and advice. It would not be an overstatement to say that they have greatly influenced me to be the person I am today. I have learnt a lot and grown a lot with my friends. My deepest gratitude goes to my family for supporting me with their love and support during some of the most difficult times. I am eternally indebted to my parents for enabling me to come to Japan and making me the person I am today.

I would also like to acknowledge the Japanese Government (Monbukagakusho, MEXT) Scholarship for funding my time in the university as a research student and during my Master's program. Similarly, I would like to thank Japan Society for the Promotion of Science (JSPS) for funding my PhD through the DC1 fellowship.

Contents

List of Figures

List of Tables

Abbreviations

AI Artificial Intelligence.

APM Adaptive Power Manager.

DiRL Distributed Reinforcement Learning.

DL Deep Learning.

DQN Deep Q-Network.

DRL Deep Reinforcement Learning.

DVFS Dynamic Voltage Frequency Scaling.

EHWSN Energy Harvesting Wireless Sensor Node.

ENO Energy Neutral Operation.

GPU Graphics Processing Unit.

IoT Internet of Things.

M2M Machine-to-Machine.

MDP Markov Decision Process.

ML Machine Learning.

MOMDP Multi-Objective Markov Decision Process.

MOO Multi-Objective Optimization.

MORL Multi-Objective Reinforcement Learning.

NN Neural Network.

QoS Quality of Service.

RFID Radio-Frequency Identification.

RL Reinforcement Learning.

RMS Root Mean Square.

SORL Single-Objective Reinforcement Learning.

TD Temporal Difference.

TENP Terminal Energy Neutral Performance.

WSN Wireless Sensor Network.

Chapter 1

Introduction

1.1 Background: The Internet of Things (IoT)

The Internet of Things (IoT) was first coined by Kevin Ashton in 1999. He was proposing to integrate the then new Radio-Frequency Identification (RFID) technology with the Internet and improve the supply chain productivity [Ashton et al., 2009]. Since then IoT has grown into a vast network connecting billions of humans and machines. It has been heralded as the second Industrial Revolution [Martin, 2014] and has already made significant impacts in many areas such as surveillance, healthcare, transportation and environmental monitoring [Alippi et al., 2009, Capella et al., 2013] to name a few. Around 2011, when it became gained massive mainstream attention, it was projected that we would have 50 billion connected devices in the IoT by 2020 [Evans, 2011]. Although we have fallen short of that goal with around "only" 26.3 billion devices connected in 2020 [Cisco, 2020], more recent estimates put us at 50 billions devices at the end of 2030 [Webb and Hatton, 2020].

The IoT enables machines to connect to each other and thus generate and process data without requiring a human in the process. These Machine-to-Machine (M2M) connections account for half of all the connected devices in IoT and are the fastest growing sector, projected to reach 14.7 billion connections by 2023 [Cisco, 2020]. They have a wide variety of pervasive applications in areas like smart agriculture, industrial monitoring and control, smart buildings and cities etc. There are millions

of such applications, each with its own service requirements and constraints on energy, computation and communication resources.

Sensor devices constitute a large fraction of the M2M connections and are indispensable for the IoT ecosystem. They operate at the very edge of the IoT - the boundary between the physical world and the Internet. They form what could be called a "digital nervous system". These devices interact with the physical environment to gather a diverse array of information in the form of images, audio, temperature, pollution levels, etc. They are typically powered by batteries and communicate through wireless protocols. Hence, they can be deployed into networks covering a wide area to gather data. Wireless Sensor Networks (WSNs) have shown great success in areas that require untethered operation like the monitoring of animal habitats [Mainwaring et al., 2002], environment [Werner-Allen et al., 2006] and civil structures [Chebrolu et al., 2008, Lee et al., 2006]. Traditionally, since they were powered by batteries only, they had a limited energy reserve and therefore a finite lifetime. As a result, they had to be optimized for low-power operation and high energy efficiency. This usually involved a tradeoff between under-utilization of the node and longer lifetimes. This also meant that they required periodic replenishing and replacement of batteries. This makes it impossible to scale the network to billions of nodes. Even if we assume optimistically that a sensor node can run on a battery for 10 years, when we consider that there will be trillions of such nodes, we are looking at 275 million battery replacements *per day*. This is not only practically infeasible but also expensive and environmentally unsustainable. Furthermore, battery energy densities have not been increasing at a rate so as to keep up with other computing technologies like processor speed, memory size etc [Paradiso and Starner, 2005]. In addition, some nodes may be deployed in hard to reach areas. Accessing them for regular maintenance may not be feasible, economically and practically.

Untethered long-term operation of WSNs requires an *inexhaustible* energy source in addition to wireless connectivity. Energy Harvesting Wireless Sensor Nodes (EHWSNs) integrate WSNs with an energy-harvesting module that extracts energy from its ambient environment. These modules can harvest energy from different sources such

as solar, wind, mechanical vibrations/pressure, temperature variations etc. Coupling energy-harvesters with an energy buffer could potentially power the sensor nodes perpetually thus enabling true autonomous operation. This potential has been realized to some degree with the rapid developments in energy-harvesting technologies. As a result, EHWSNs have had enormous success in a variety of applications such as animal monitoring, personal health care, disaster prevention, etc. EHWSNs show great promise in realizing an autonomous and perpetual network of sensors for IoT.

1.2 Energy Neutral Operation (ENO)

Traditionally, WSNs were optimized for low-power operation [Sinha and Chandrakasan, 2001, Min et al., 2000] or to maximize their lifetimes [Singh et al., 1998, Younis et al., 2002, Shah and Rabaey, 2002]. However, with the introduction of EHWSNs, the nodes had access to potentially infinite energy, although at a finite (and typically low) rate. One could over-provision EHWSNs so that it always scavenges more energy than it consumes and equip it with a large storage buffer. However, this drives up the cost and the form factor of the sensors without any increase in utility. Another straightforward solution to maximize the node lifetime would be to simply recharge the batteries as they ran low. While this ensures long-term operation, the nodes still operate so as to minimize energy consumption and are therefore under-utilized. In [Kansal et al., 2007], the authors proposed a radically different problem formulation by introducing the concept of Energy Neutral Operation (ENO). The main intuition behind ENO is that the nodes could harvest and store surplus energy to be used at a future time to extract more utility. This resulted in a fundamental shift in the optimization paradigm from reducing device consumption to optimal energy scheduling. Energy Neutral Operation (ENO) meant that we could extract maximal utility from the node while ensuring long-term uninterrupted operation. This was first formally presented in [Kansal et al., 2007] which sparked a flurry of papers on optimizing EHWSNs to maximize *the utility of the node* with intelligent energy management policies.

However, energy neutral policies that can sustain long-term operation while meeting user-defined performance requirements is not easy to design. Sensor nodes interact with the real physical world which is an open system that is dynamic and ever changing. The primary difficulty arises due to the uncertainty of the energy-availability from the environment. Harvested energy is typically limited, unpredictable and unreliable. For instance, solar energy, which is one of the most popular energy-harvesting technique for EHWSNs, fluctuates throughout the course of the day and with different weather and climate conditions. It is very difficult to predict and can vary rapidly. This is even more severe when we consider mobile sensor nodes [Zhang et al., 2004]. Other energy sources such as wind, vibration are also equally unreliable. If we could harvest this energy at any arbitrary rate and store it in an infinite buffer, energy management would not be a difficult issue. However due to the finite harvesting rate and the limited battery capacities, we need intelligent policies that can schedule the energy consumption for optimality. Many analytic solutions have been proposed to deal with this resource allocation under uncertainty using methods like linear programming [Kansal et al., 2007], non-linear programming [Peng and Low, 2014, Cionca et al., 2018], search methods [Jia et al., 2019], control systems [Vigorito et al., 2007], dynamic programming [Fu et al., 2006, Lei et al., 2016] among other methods. One method is to use prediction mechanisms that can provide approximations of non-causal information to carry out the optimization process. Solar energy predictions play a central role for solutions in [Kansal et al., 2007, Geissdoerfer et al., 2019, Buchli et al., 2014, Cionca et al., 2018]. On the other hand, prediction-free solutions [Peng and Low, 2014, Vigorito et al., 2007] optimize on the basis of some user-defined parameters and constraints.

In addition to energy variability, there are also other external factors that the energy policies must compensate for. This includes changes in energy harvesting efficiency, battery degradation [Michelusi et al., 2013] and decrease in node's energy efficiency. Considerations must also be made for the utility of the sensor node w.r.t. user requirements. For instance, EHWSNs that maximize communication-based metric such as throughput or latency must optimize their operation in the face of un-

predictably varying channel conditions such as noise, congestion and so on. Modern sophisticated EHWSNs are typically required to execute a variety of tasks related to sensing, communication and processing [Ma et al., 2019, Nakamura et al., 2017]. In such a case, the node needs to optimally allocate the energy between different tasks to maximize its utility. It is neither easy nor trivial to express all these numerous factors into a single optimization problem statement; and actually solving it is often very difficult. As a result, most solutions usually make strong assumptions to simply the problem and develop tractable solutions. Consequentially, these solutions have a very limited scope of application and have strong implicit design bias. For instance, the solutions in [Ozel et al., 2011, Sharma et al., 2010b] only maximize throughputs and cannot be used for optimizing sensing rates like in [Dias et al., 2016]. Some solutions make very strong assumptions such as infinite energy buffers [Reddy and Murthy, 2012]. Almost no analytic solutions exist that take into account the long-term degradation of the sensor node. One could attempt to integrate adaptive heuristic policies to account for the variations in the working environment. However, this rapidly compounds the complexity of the optimization problem and is not a practical solution. Another alternative would be to schedule routine maintenance and fine-tuning of the sensor nodes and energy policies. This is also not a practical solution due to the massive scaling in the number of the sensor devices and defeats the idea of autonomous long-term operation of sensor nodes.

The scalability of the energy policies is another critical factor that needs to be considered for EHWSNs now that we are dealing with billions of such sensor devices. We need to consider policies that can scale not only with the *number* of sensors but also with the *diversity* of their applications. Present day IoT consists of heterogeneous networks, devices and systems that are interconnected and working together. There are millions of different application scenarios that may employ a variety of different sensors with a multitude of energy-harvesting scenarios. Analytic solutions are typically application-specific and need to be recalibrated for different scenarios. Manually calibrating each sensor device for optimality is just not possible at the present scale. As the number and usage scenarios of EHWSNs increase each day, a one-size-fits-

all energy management framework based on analytic methods seems more and more improbable.

Heuristic methods are non-adaptive, application-specific and require non-causal information and significant manual calibration. These limitations have restricted the use of EHWSNs in research-centric niche areas. We need to overcome these challenges so that EHWSNs can transition to mainstream applications in industry and daily life. What we urgently require is a general and unified energy management solution that can be modified for different EHWSNs and their usage scenarios with minimum human intervention. The policies should be able to ensure ENO for long-term operation and adapt to changes in their working environment, the device parameters and the performance requirements.

1.3 Self-Adaptive Policies and their Challenges

1.3.1 Reinforcement Learning (RL)

Within the past decade, learning-based methods have risen as a popular alternative to analytic methods for ENO. Machine Learning (ML) can be broadly categorized into supervised learning, unsupervised learning and Reinforcement Learning (RL). While supervised and unsupervised learning are typically concerned with classification and regression problems, RL is most suited for sequential decision-making tasks [Sutton and Barto, 2018]. In this work, we focus on RL algorithms to achieve ENO in EHWSNs.

RL is a learning paradigm based on instructive feedback instead of evaluative feedback. It finds its roots in the psychology of animal learning and dynamic programming [Bellman, 1954]. RL can be thought of as an amalgamation of trial-and-error learning, optimal control and temporal difference learning. RL agents learn through direct interaction with an environment to achieve long-term goals. They learn to make decisions under uncertainty without requiring explicit supervision and complete models of the environment. The general learning paradigm is based on a perception-action-

reaction feedback loop. The agent observes the *state* of its environment and executes an *action*. The environment reacts to this with a *reward* signal and a change in its state. The reward feedback represents the optimality of its actions w.r.t. its objective.

Energy scheduling for ENO requires decision-making under uncertainty. This make RL very suitable for EHWSNs and it overcomes many of the limitations of analytic methods. The essence of RL lies in the fact that the agent can automatically learn policies to accomplish a certain goal without any instructions from the user on how to do it. The agent is expected to learn via trial-and-error. Of course, the agent will behave sub-optimally during its learning phase. However, we can expect it to behave optimally and get better with more experience. This frees the user from having to specify separate policies for different application contexts as is usually done with heuristics. In fact, it may even be possible for the agent to learn novel policies that outperform the ones designed by humans.

RL requires the designer to specify the problem formulation in the form of a Markov Decision Process (MDP). The user specifies the rules of the interaction with the environment by defining the state variables, the action space and the reward function. With suitable RL algorithms, the node then can learn optimal policies depending upon the desired objective. The user does not need to specify application-specific parameters and constraints such as those required by analytic methods. As a result, the same general problem formulation can be used in a variety of sensor nodes and usage scenarios with only minor changes. This makes RL-based optimization applicable in a wide variety of working environments and therefore extremely scalable.

Another important advantage of RL for ENO is the adaptive nature of learning. Since the agent learns via interactions, it follows that the agent can learn to adapt accordingly if the environment should change. Variations in its working environment affect the rewards obtained by the agent which is then used to modify the policy accordingly. This adaptive behavior applies not only to the changes in external environment but also to changes in the device parameters as a consequence of long-term operation. These include degradation of battery efficiency and capacity, node inefficiencies, channel congestion etc. These self-adaptive policies play an ex-

tremely critical role in realizing truly autonomous sensor nodes which can be used in deploy-and-forget scenarios.

RL methods overcome many of the limitations of analytic methods for ENO. They are general, self-adaptive, scalable and can learn to make optimal decisions in the face of uncertainty. As a results, sensor nodes will be capable of autonomously learning optimal strategies and adapting once deployed in their working environment. So it should not come as a surprise that there has been a massive interest in using RL for IoT devices in the past few years [Luong et al., 2019, Ma et al., 2019, Jagannath et al., 2019]. They have been used for adaptive rate control [Hsu et al., 2014, Shresthamali et al., 2017, Dias et al., 2016, Murad et al., 2019, Shresthamali et al., 2019, Xu et al., 2018, Xu et al., 2019], communication optimization [Blasco et al., 2013, Ortiz et al., 2016, Ortiz Jimenez, 2019, Qiu et al., 2019, Kim et al., 2019, Van Huynh et al., 2019, Long and Büyüköztürk, 2020], routing [Mammeri, 2019] in EHWSNs.

1.3.2 Challenges in RL for ENO

Although RL is extremely versatile and powerful, it comes with significant challenges and limitations. Some of these limitations are due to the nature of RL while the others arise because EHWSNs interact with the physical world which introduces significant uncertainty. We discuss some of these limitations below.

Lack of strong guarantees

RL is a relatively new field that still has many open challenges that need to be addressed. Most theoretical guarantees for RL are made only under very strong assumptions. In spite of this, RL has already proved its potential in a variety of practical applications. However, most of them are simulation-based and have very controlled deterministic environments [Silver et al., 2016, Mnih et al., 2015, Vinyals et al., 2019]. It is still very difficult to give performance guarantees for RL in the real world except for very simple scenarios. One of the reasons for this is because the learning process is inherently stochastic. RL learns via bootstrapping where policies

generate experiences which in turn are used to update the policies. This makes it very difficult to reproduce results and convergence guarantees cannot be made for most practical scenarios. If one is not careful during the learning process, stochastic variations can sometimes break the RL solutions.

Problem Definition

RL typically involves first defining the RL problem as a MDP and then employing a suitable RL algorithm to learn optimal policies. Defining the MDP involves specifying the state variables, the action space and the reward function. The state variables define what information from the environment is used to make decisions. For e.g., in EHWSNS, state variables may include information about harvested energy levels, battery reserves, channel congestion etc. The action space defines the actions that the agent can choose from, e.g., node duty cycles, transmission bitrate, sensing rate etc. The reward function defines the rewards the agent receives for its chosen action. The reward reflects the ultimate goal of the agent. For e.g., high rewards may be awarded for high duty cycles or throughputs whereas negative/low reward may be given if the battery is depleted and the node has to be turned off. It is only on the basis of these rewards that the agent can learn what is the optimal policy that maximizes its *cumulative* reward in the long run.

Defining the state-action space is not always obvious and can affect the learning process significantly. For instance, using discrete definitions for state-action spaces may reduce the complexity of the learning process as in the case of tabular RL. However, this greatly diminishes the scope of application as compared to an MDP with continuous state-action spaces. Determining the state variables is also not that easy because it is also always not obvious what information is pertinent when making decisions.

More importantly, defining the reward function is a very difficult issue when it comes to ENO. Rewards reflect the true objective of the agent and it is difficult to express this when we have multiple conflicting objectives and different optimization horizons. Sometimes, it is not even clear what the reward should be. For instance,

when maximizing the throughput of the node, should the reward be a binary signal that represents transmission success or should it be a continuous value that represents the throughput? This issue becomes even more contentious when we have to optimize over multiple objectives.

Learning Costs

Learning costs refer to the loss of utility incurred by sub-optimal policies during the learning process (e.g., node shutdowns, energy wastage) as well as the time required by the agent to converge to a working near-optimal policy. RL learns via trial-and-error and mistakes are inevitably necessary during the initial stages of training. Obviously, we want the learning period to be as short as possible so as to maximize utility immediately. Additionally, we want to learn with minimal mistakes because they may have potentially catastrophic consequences in during real-world deployment, e.g., node shutdown due to energy depletion. However, RL algorithms have notoriously high learning costs and they grow larger as the problem becomes more complex. This becomes a very contentious issue when we combine Deep Learning (DL) with RL as a result of its low sample-inefficiency.

Computation Costs

Although modern EHWSNs [Kinney et al., 2003, Nakamura et al., 2017, Libelium, 2021] have substantial processing power, it is still a long way from being able to implement the powerful algorithms required for Deep Reinforcement Learning (DRL). Most practical solutions are limited to tabular RL methods [Fraternali et al., 2019] or very application specific custom hardware [Restuccia and Melodia, 2020]. Nevertheless, limited memory and computation resource as well as energy constraints make it very difficult to implement learning algorithms on the nodes.

1.4 Contributions

In this work, we propose solutions to overcome some of the challenges of using RL for ENO. For the sake of example, we simulate a solar-powered EHWSN to test our RL solutions. Solar EHWSNs are the most popular and widely used sensor nodes. They present significant energy variability which makes the learning process quite challenging.

The basic system model is summarized in Figure 1-1. A finite rechargeable battery is used as an energy buffer. The sensor node is capable of varying its energy consumption via duty cycling. The RL power manager determines the optimal duty cycle to maximize the utility of the node while ensuring long-term ENO.

Our work focuses on the following directions to overcome the challenges of using RL for ENO:

- First we investigate the possibility of a unified general learning framework for EHWSNs that is independent of the specifics of the working environment. As a proof of concept, we verify if RL is capable of learning convergent and adaptive policies in this framework?
- Second, we investigate a distributed approach on how to reduce the learning costs associated with RL without compromising on the optimality and robustness of the learned policies?
- Finally, we look into how we can extend our RL framework to multiple objectives so as to allow for dynamic runtime tradeoffs between different conflicting performance demands?

1.4.1 General Framework for Adaptive Policies (Chapter 4)

For the first contribution, we first look into the feasibility of RL for ENO and investigate its adaptive behavior. Many of the existing RL solutions for ENO usually optimize for some communication-centric metric. These solutions require different reward functions depending on the application and context. This limits the generality

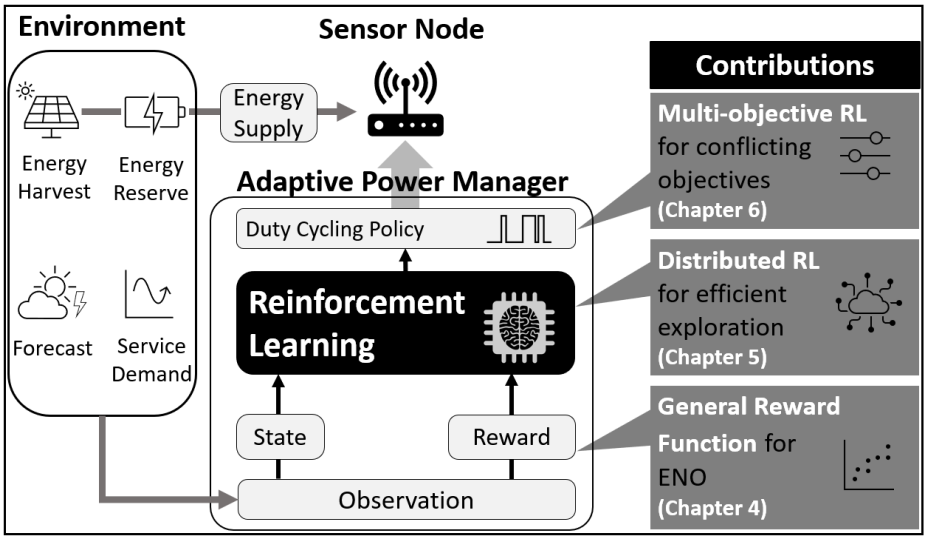

Figure 1-1: We simulate a solar-powered EHWSN as a testbed for evaluating our RL solutions. Our work focuses on developing a RL framework with which we can derive policies for a wide variety of EHWSN application scenarios.

of their problem formulation. Moreover, very few solutions exist that explicitly optimize for ENO and they require significant reward-shaping. So we develop a simple but general/unified EHWSNs system model and propose a novel MDP formulation for RL. This formulation includes a novel specification of a general reward function that directly reflects the ENO objective so that it works for most ENO applications irrespective of the application context. However, learning with such a general reward function is not easy and may diverge. Thus we use the tabular SARSA(λ) RL algorithms to make such a general reward function feasible. Our solution uses tabular RL to maximize the duty cycles of the sensor nodes. We demonstrate that our RL solution is convergent and capable of adaptation to environment and different device parameters. This demonstrates that our RL formulation can be used for a wide variety of application scenarios. Furthermore, tabular RL requires minimal computational resource once the training is finished which makes it feasible for real-world deployment. This has been successfully verified in [Fraternali et al., 2019] where the authors use a very similar approach to ours. Thus, for our first contribution, we establish that RL methods are a feasible alternative to heuristics and could adapt to changes in the working environment. A major part of this contribution has been

published in our paper - [Shresthamali et al., 2017].

1.4.2 Coordinated Exploration to Accelerate Learning (Chapter 5)

Tabular RL requires discretization of states and actions which limits the generality and scalability of the solution. To overcome this, we use Deep Q-Networks (DQNs) that can deal with continuous states and discrete actions. DQNs use Neural Networks (NNs) as function approximators to learn powerful energy neutral policies. They are more general and powerful than tabular solutions. However, they are also notoriously sample-inefficient and require significant computing power during the learning process. We identify inefficient exploration as one of the major sources of high learning costs associated with DQNs. Insufficient exploration leads to weak sub-optimal policies. On the other hand, robust policies require long exploration periods which drive up the learning costs. Learning costs include the time spent during training and the loss of utility due to sub-optimal actions during this period. So for our second contribution, we propose a Distributed Reinforcement Learning (DiRL) solution to coordinate the many nodes of the sensor network to efficiently explore the vast state-action space so that the robust policies can be learned in a short period of time. We propose efficient exploration strategies for DiRL to learn near-optimal energy neutral policies while trading off between learning costs and the robustness. Additionally, we also off-load all learning computations to a powerful central server thus greatly reducing the computation burden on resource-constrained EHWSNs. The learning framework is distributed in the sense that multiple agents interact simultaneously with the environment to gather experience. However, it is also not strictly *distributed* in the sense that all learning is performed on a single powerful computer (centralized learning). In our case, we assume that this computer is housed in the WSN base station without any strict restrictions on its compute and energy resources. Our method was able to decrease the learning period from years to a matter of days using our distributed framework and coordinated exploration approach. This solution was

published as [Shresthamali et al., 2019].

1.4.3 Optimizing over Multiple Objectives (Chapter 6)

Traditionally, EHWSNs were simpler and it was sufficient to maximize a *single* metric such as the communication throughput [Ozel et al., 2011, Ortiz et al., 2016, Aoudia et al., 2018] or the sensing rate [Vigorito et al., 2007, Dias et al., 2016, Murad et al., 2019]. This has also been the underlying assumptions in the previous contributions. However, modern EHWSNs are more sophisticated and are typically required to execute a variety of tasks related to sensing, communication and processing [Ma et al., 2019, Nakamura et al., 2017]. In such a case, the node needs to optimally allocate the energy between different tasks to maximize its utility and maintain ENO. There are multiple task objectives that need to be optimized simultaneously, but since all tasks draw energy from the same limited source, tradeoffs need to be made based on user-defined preference or *priority*. It is therefore necessary to develop an intelligent energy management strategy that ensures ENO and optimizes tradeoffs between different tasks. This naturally points to a multi-objective optimization approach.

Analytic methods for multi-objective optimization usually rely on non-causal, a priori information and application-specific parameters to construct a solution. They require significant user bias and grow very complex as more tasks are added. Their limited scope and adaptivity is a serious limitation. On the other hand, most RL methods are not suitable when multiple objectives are concerned primarily because they are based on learning from a single *scalar* feedback reward signal. When multiple objectives are concerned, they are usually projected onto a scalar [Ferreira et al., 2018, Aoudia et al., 2018, Hsu et al., 2014, Murad et al., 2019]. This approach is limited because tradeoffs cannot be made and it requires a specialized scalarization function with *prior* knowledge of the priorities.

Thus, for our final contribution, we propose a novel and general framework for multi-objective optimization of EHWSNs using Multi-Objective Reinforcement Learning (MORL) [Vamplew et al., 2008]. Our MORL framework consists of a novel Multi-Objective Markov Decision Process (MOMDP) formulation and two MORL

algorithms. Our framework facilitates the learning of intelligent policies with very low learning costs, that tradeoff over multiple objectives *a posteriori* using a *vector* of reward signals. Our *general* energy management framework can dynamically optimize the energy usage among *multiple objectives* in a realistic and feasible manner. We show that this solution can learn superior policies with much lesser learning costs and are capable of dynamic tradeoffs within acceptable computation costs.

1.4.4 Summary

We summarize the contributions of our work in Figures 1-2 to 1-4. Figure 1-2 shows how our proposed contributions are advantageous over traditional heuristic methods. By using RL-based methods, we can derive adaptive policies without having to customize each solution depending on the working environment. Instead, we use a common learning framework which require minimal design bias that can generate convergent policies for a wide variety of application scenarios.

One of the major objectives of our work is to reduce the learning costs associated with RL. Naive implementation of existing algorithms generally result in large learning costs. This is mainly due to improper MDP specifications, poor exploration strategies and improper RL algorithms. Our contributions decrease the learning costs w.r.t. naive RL implementation as summarized in Figure 1-3. In our first contribution, we decrease the learning costs by using an improved MDP and substituting the offline tabular Q-learning with the online SARSA(λ) algorithm. In our second contribution, we were able to dramatically reduce the learning costs by using coordinated exploration strategies. For our third contribution, we modify the RL problem formulation and MDP to include multiple reward signals. By adding more temporal information into the state space and using *relative* actions (as opposed to absolute ones), we were able to accommodate multiple rewards while decreasing the associated learning costs.

As discussed before, the computation costs of RL methods makes their applicability in EHWSNs very challenging. Although reducing computation costs is not the primary focus of our work, we have tried to minimize the computational overhead

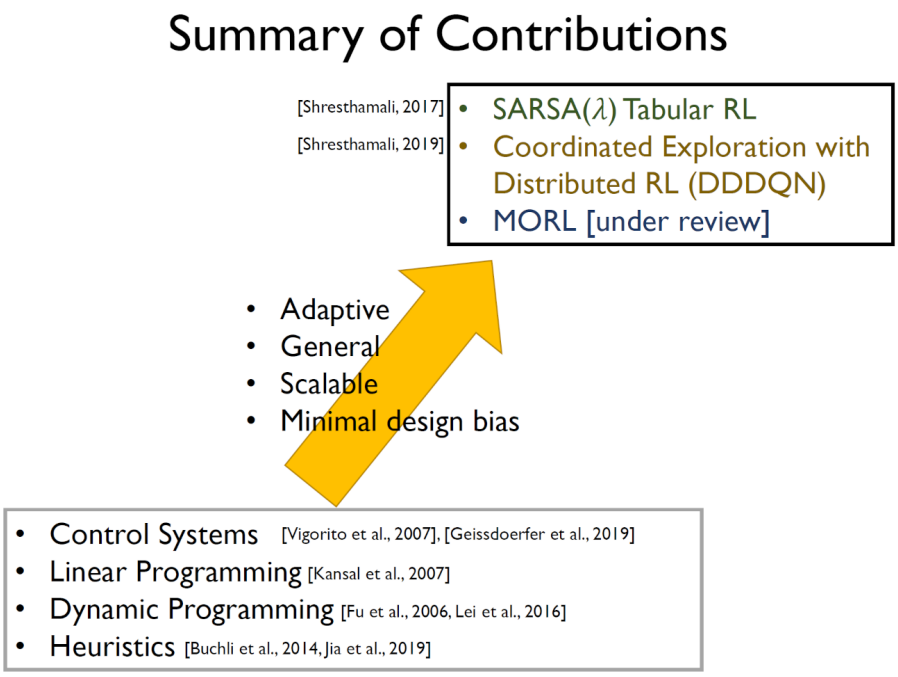

Figure 1-2: This figure summarizes the main contributions of this work and compares it to existing analytic methods. We show that by using RL in its various forms - tabular (Chapter 4), NN-based function approximation (Chapter 5) and multi-objective optimization (Chapter 6), we can generate general, scalable and adaptive energy neutral policies that require minimal hand-tuning.

whenever possible. This is illustrated in Figure 1-4. Our proposed tabular method (SARSA(λ)) in Chapter 4 has the least compute requirements (mostly required during the learning process). It is only slightly more expensive than traditional analytic methods but is capable of learning near-optimal energy neutral policies. Once the policies have been learned, inferring the optimal policy is actually very cheap. DQN methods [Mnih et al., 2013, Mnih et al., 2015, Hessel et al., 2018] are even more powerful but they require extremely large computing resources (multiple CPUs and GPUs). When we use distributed learning as in [Horgan et al., 2018, Nair et al., 2015], the computing costs get compounded. Our proposed distributed learning framework concentrates all the computation costs onto a central server which minimizes the computation loads on the sensor nodes thus greatly reducing the compute requirements in sensor nodes. The sensor nodes execute only NN inference tasks which is much cheaper. As a result, our proposed solution in Chapter 5 produces more robust

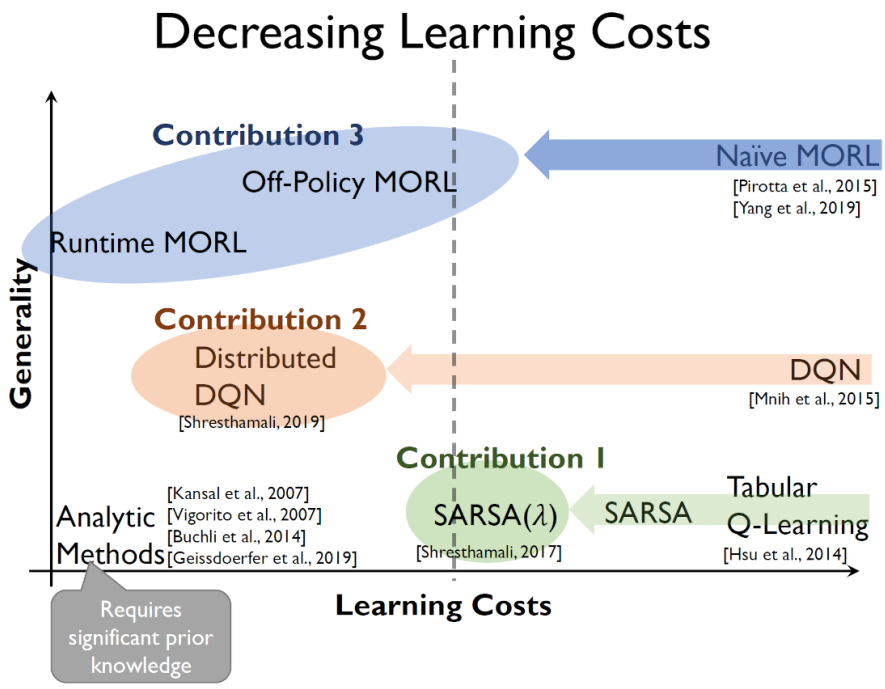

Figure 1-3: Since RL is a trial-and-error based learning paradigm, it is inevitable that there will be some non-optimal behaviour in the beginning of the learning phase. However, for real-life scenarios such as in EHWSNs, we would like the agents to learn quickly from its mistakes while making as few of them as possible i.e., decreasing its learning costs. Our proposed methods decrease the time and risks associated with learning without compromising on the generality of the approach as compared to naive RL implementation as illustrated in the figure.

and optimal policies with reduced compute requirements. Similarly, direct MORL methods [Pirotta et al., 2015, Parisi et al., 2014, Yang et al., 2019] also require large compute resources since multi-objective optimization using RL is a much harder and complex problem. In Chapter 6, we propose two novel MORL algorithms - Runtime MORL and Off-Policy MORL - to reduce the computation costs associated during the learning and inference process.

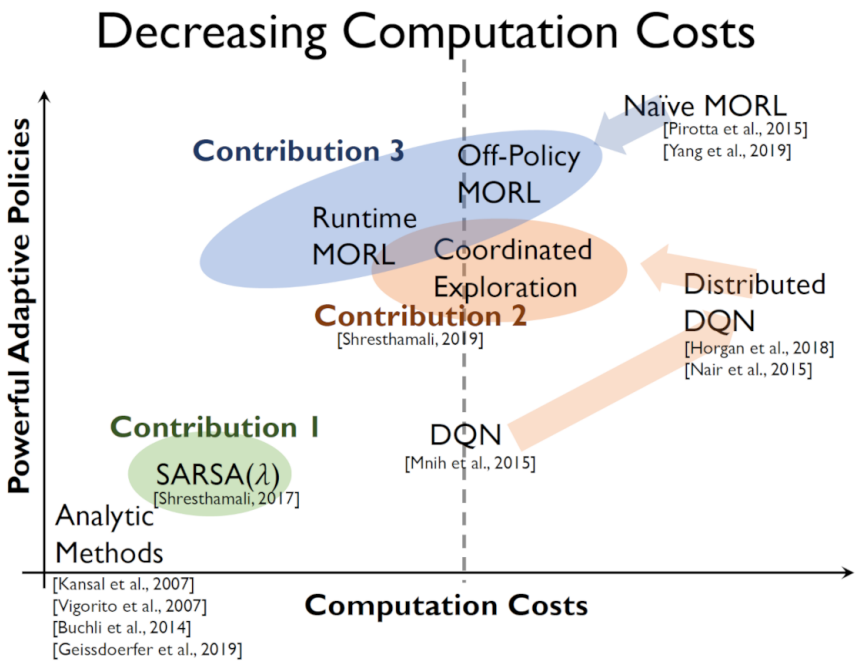

Figure 1-4: The main motivation behind this work has been to identify the limitations of using RL for implementation in EHWSNs. The figure illustrates how our novel algorithms and problem formulations can generate powerful policies without requiring the huge computation costs that are usually associated with naive RL algorithms.

1.5 Organization

We organize this dissertation as follows. Chapter 2 reviews relevant literature and we show how our work contributes to the missing fields in the area. Chapter 3 lays down the fundamental theory related to ENO and RL. Chapter 4 goes into depth about the general MDP formulation for adaptive learning and discusses the evaluation results. Chapter 5 explains our novel exploration strategies for efficient learning using DiRL and the results of our evaluations. In Chapter 6, we introduce the MORL framework and evaluate the proposed MORL algorithms. We conclude with Chapter 7 and follow up with some notes on the limitations of this work and future research directions.

Chapter 2

Literature Review

2.1 Analytical Methods for ENO

Historically, WSNs were powered only by batteries which meant that they were optimized for minimal power consumption so as to maximize their lifetimes [Sinha and Chandrakasan, 2001, Min et al., 2000, Singh et al., 1998, Younis et al., 2002, Shah and Rabaey, 2002]. This usually involved optimizing the entire network holistically [Mills, 2007] and mainly focused on optimal routing strategies [Al-Karaki and Kamal, 2004, Kalantari and Shayman, 2004, Schurgers and Srivastava, 2001, Huang and Tseng, 2005], event-driven sensing [Sundaresan et al., 2009] or novel physical layer protocols [Shih et al., 2001]. A general overview of the design and working of wireless sensor networks is discussed in [Yick et al., 2008]. A discussion of WSNs on the basis of their energy provision and consumption is given in [Khan et al., 2015].

With improvements in low-power electronics, wireless standards and sensor miniaturization, the idea of integrating energy-harvesters with sensors took off around 2005. In [Paradiso and Starner, 2005], the authors present different energy-harvesting methods and their challenges in real-life application scenarios. Soon goals of long-term operation of sensors were being proposed. In [Jiang et al., 2005], the authors present the design and implementation of *Prometheus* - a long duration solar power subsystem for a wireless sensor node. The design challenges and possible solutions for similar systems are summarized in [Gunduz et al., 2014]. Not soon after, the concept

of ENO was formulated and described in a series of papers by Kansal et al [Kansal et al., 2007, Kansal et al., 2004, Raghunathan et al., 2005]. This set the foundation for research on optimization techniques for EHWSNs in different applications and environments [Alippi et al., 2009]. Different energy harvesting architectures and real world examples of EHWSNs are surveyed in [Sudevalayam and Kulkarni, 2011, Warrier and Kumar, 2016, Fei et al., 2016]. Energy aware security in WSNs is discussed in [Di Mauro and Dragoni, 2015].

ENO of EHWSNs have mainly focused on maximization of either the gross device energy consumption [Kansal et al., 2007, Vigorito et al., 2007] or the communication performance [Sharma et al., 2010b, Ozel et al., 2011] using linear programming [Kansal et al., 2007], non-linear programming [Peng and Low, 2014, Cionca et al., 2018], search methods [Jia et al., 2019], control systems [Vigorito et al., 2007], dynamic programming [Fu et al., 2006, Lei et al., 2016] among other methods [Gregori and Gómez-Vilardebò, 2016, Wang et al., 2015a, Jiang et al., 2005, Gunduz et al., 2014]. The research community especially focused on the network performance of EHWSNs [Ulukus et al., 2015, Lu et al., 2015] . The authors in [Dinh et al., 2016] were able to propose a global power management framework for such cases.

Solar cells have a comparatively high energy density compared to other energy scavenging methods. In addition, their energy production profile is also predictable to some degree due to their quasi-periodicity. As a result, solar powered wireless sensor nodes have been extensively researched in many real-world application areas. In [Alippi et al., 2011, Capella et al., 2013], the authors use a solar powered WSN for aquatic environmental monitoring; greenhouse gases monitoring using solar powered aerial vehicles is presented in [Malaver et al., 2015]; structural health of bridges have been monitored using solar powered WSNs in [Cho et al., 2010]. In [Wu et al., 2017] report on the performance of an IoT EHWSN in real-world scenario. To this end, we use a solar energy-harvesting scenario for our research.

Our research is concerned with a scalable and general energy management solution. In this regard, we build our research on the mathematical framework and objectives of ENO laid by [Kansal et al., 2007]. The authors propose a linear programming

optimization solution that leverages non-causal information about the environment. They predict the amount of energy that is expected and decide on the duty cycle accordingly. They take into account battery inefficiencies and also allow for adaptation of the duty cycle to accommodate for any changes in predicted and actual harvesting conditions. However this technique relies heavily on the accuracy of its prediction mechanism and prior knowledge of the statistics of the environment.

In [Vigorito et al., 2007], the authors approach the problem of energy neutrality by specifying a battery-centric objective function that does not require any non-causal information. They restate ENO as a control system problem that minimizes the deviations in the battery level and achieve energy neutrality by using a linear quadratic-tracker. They also take into account the minimization of duty cycle variance. They do not attempt to model the energy source and do not require prior information about the environment. While this approach is adaptive in nature, it requires careful calibration of hyper parameters depending upon the working environment. The agent would not be able to autonomously "fix" its own hyper parameters if its environment should change. Similar control system solutions are also presented in [Buchli et al., 2014, Geissdoerfer et al., 2019]. In [Geissdoerfer et al., 2019], the authors optimize the operation of EHWSN while taking into account the *dynamic utility* of the data gathered by the sensor node.

2.2 Learning-Based Methods for ENO

While EHWSNs were getting more sophisticated, data-driven Machine Learning (ML) for Artificial Intelligence (AI) was also getting renewed interest mainly due to the massive data generated by the Internet. The availability of cheap data facilitated the research of AI technologies that had been dormant since the 1940s. The most notable milestones were the development of the back-propagation algorithms [Rumelhart et al., 1986] coupled with Graphics Processing Unit (GPU) to accelerate the learning in Neural Network (NN) architectures [Krizhevsky et al., 2012]. This led to the development of Deep Learning (DL) which has since then grown with extreme rapidity. It

has had a significant disruptive effect in industry and research alike. DL algorithms are data-driven methods to train NN that have multiple hidden layers. These NN can be used to approximate complex non-linear functions that can be used for different purposes such as regression, classification, representation learning, feature extraction etc. DL algorithms have significantly enhanced the capabilities of all of these ML branches. More recently, DL have enabled powerful function-approximation for RL giving rise to Deep Reinforcement Learning (DRL).

With the success of machine learning, it was soon applied to overcome various challenges of WSNs[Alsheikh et al., 2014]. RL methods for power management of WSNs are discussed in [Rucco et al., 2013, Yau et al., 2012]. We were also inspired by the RL algorithms used in [Wang et al., 2011] for power management of hard disks and Wireless Local Area Network (WLAN) cards; in [Wei et al., 2017] for heating, ventilation and air-conditioning; in [Lee and Powell, 2012] and [Ermon et al., 2012] for intelligent battery usage policies. Many RL solutions already exist that optimize traditional non-energy-harvesting WSNs. They usually focus on optimizing the performance of the *network* rather than individual nodes [GhasemAghaei et al., 2007, Wang and Wang, 2006, Liu and Elhanany, 2006].

Early applications of RL in EHWSNs concentrated on optimizing communication policies [Blasco et al., 2013, Ortiz et al., 2016, Jia et al., 2019]. In [Blasco et al., 2013], the authors tackle the case of point to point wireless communication system with stochastic data arrival and channel state processes. They provide optimization techniques when non-causal information about the environment is known and in cases when statistics of the environment are known beforehand. They then propose an RL based learning theoretic approach to maximize total transmitted data. The authors use discrete states, actions and rewards similar to our approach. In [Ortiz et al., 2016], the authors improve on this by using linear function approximation with RL to accommodate for continuous states and rewards. [Mao et al., 2014, GhasemAghaei et al., 2007, Blasco and Gündüz, 2015, Chan et al., 2015, Xiao et al., 2015, Mihaylov et al., 2009] also focus on optimizing power consumption of EHWSNs using RL under the assumption that communication operations consume the most significant amount

of power. This is not always a reasonable assumption in general. For instance, mobile sensor nodes consume significant energy in locomotion as compared to sensing and communication operations [Howard et al., 2002a, Rahimi et al., 2003, Howard et al., 2002b, Verma et al., 2006, Kim and Chung, 2006]. A slightly different problem was tackled in [Chan et al., 2015] where duty cycling methods using continuous time Markov Chain Modeling by taking into account the Quality of Service (QoS) requirements and the battery chemistry.

A general RL solution that optimizes the node regardless of its primary function was proposed in [Hsu et al., 2006] by optimizing the duty cycle to achieve node neutrality. This solution used a very non-intuitive and complex reward function. Their basis for reward is the distance from energy neutrality and the residual battery level. While this is a more general formulation of the reward function, it is handcrafted. The reward function should be an indication of what kind of states or actions are favorable rather than directives on *how* to achieve a certain goal. In our first contribution (Chapter 4), we build upon this and devise a simpler, general reward function capable of learning adaptive policies. None of the previous research have investigated the adaptivity of their algorithms to changes in climate, device parameters and battery degradation. In this thesis, we also analyze the adaptive behavior of our RL-based policy to various environmental changes. [Sharma et al., 2010a] show that performance of EHWSNs can be improved by using weather forecast information. Their work uses this information to make better predictions about the energy that can be harvested. Along the same lines, we also leverage the use of easily obtainable forecast data for a particular day to improve performance. Other research similar to our work [Shresthamali et al., 2017] include [Xu et al., 2019, Gai and Qiu, 2018, Xu et al., 2018], where the authors also consider Quality of Service (QoS) and task dependencies. Model-based approaches are discussed in [Braten and Kraemer, 2018].

ENO-centric RL methods using tabular methods [Blasco et al., 2013, Hsu et al., 2014, Shresthamali et al., 2017, Gai and Qiu, 2018, Xu et al., 2019, Xu et al., 2018] and in conjunction with linear function approximation [Aoudia et al., 2018, Ortiz et al.,

2016] have low computational requirements but apply only to small discrete state-action spaces. More powerful RL policies can be learned using Deep Reinforcement Learning (DRL), first introduced in [Mnih et al., 2015]. In this paper, researchers used Neural Network (NN) called DQNs to learn complex RL policies that could achieve superhuman scores in arcade video games. This demonstrated the enormous potential behind RL and brought about a new field of DRL which was a confluence of Deep Learning (DL) and RL.

However, using DRL for EHWSNs is quite challenging. After 2015, there has been many research efforts in this area [Rioual et al., 2018, Murad et al., 2019]. DQNs are known to be sample inefficient and therefore suffer from lengthy learning times. They require a memory pool from which minibatches are extracted for training. Populating this memory pool is expensive in terms of time and performance. The authors in [Nair et al., 2015, Horgan et al., 2018] propose DiRL to populate this pool via concurrent interaction of multiple agents with the environment. However since their application domain (playing computer games) and real-life EHWSNs are very different, naive implementation of their method is not optimal. Computer games are artificial deterministic environments where disastrous outcomes incur no real costs. Thus, as our second contribution (Chapter 5), we propose novel ϵ-greedy exploration strategies to not only minimize the learning time but also reduce the costs associated with learning. We would like to note that, in our work, the DiRL agents do not interact with each other, either directly or indirectly. This is not an unreasonable assumption to make for EHWSNs because harvesting ambient energy and sensing environmental data has practically no effect on the working environment. RL agents whose actions influence the environment experienced by other agents lie in the domain of multi-agent RL and is not in the scope of our research.

The exploration-exploitation dilemma in RL is still an open problem. ϵ-greedy and softmax action selection [Sutton and Barto, 2018] are popular methods of exploration via introducing noise in the action space. The author in [Thrun, 1992] gives an overview of different exploration techniques in action space. In contrast, Noisy-nets [Fortunato et al., 2017] and evolutionary strategies [Salimans et al., 2017, Khadka and

Tumer, 2018, Khadka et al., 2019] inject noise into the parameters of the neural network to achieve exploration. While the above methods tackle *how* to explore, methods based on concepts of surprise [Achiam and Sastry, 2017], curiosity [Pathak et al., 2017], selective attention [Sorokin et al., 2015], disagreement [Pathak et al., 2019] and gradient ascent in information [Houthooft et al., 2016] try to answer the question of *when* to explore. These sophisticated exploration methods, while effective in selective domains, require non-trivial computational requirements and therefore have limited application in resource-constrained EHWSNs. In the end, ϵ-greedy methods give the best bang-for-buck with good exploration at minimal computational cost over other methods. In [Masadeh et al., 2018], the authors compare convergence-based and ϵ-greedy exploration strategies for single agent RL systems for EHWSNs.

Adaptive ϵ-greedy methods have been proposed in [Tokic and Palm, 2011] where the authors propose a fusion of softmax and ϵ-greedy methods depending on the temporal difference error. However, this method is not applicable for DQNs. Our adaptive ϵ-greedy method follows similar logic but adapts the *change* in exploration rate as a function of the immediate reward. As to our knowledge, this is the first work that exploits ϵ-greedy methods for accelerating DiRL by introducing preferential exploratory actions. Our work can also be easily extended to policy gradient [Sutton and Barto, 2018, Schulman et al., 2015] and actor-critic RL [Mnih et al., 2016] methods.

2.3 Multi-Objective Reinforcement Learning (MORL)

Most research related to ENO of EHWSNs have largely concentrated on optimizing a *single* objective. For instance, analytic methods such as [Kansal et al., 2007, Vigorito et al., 2007] optimize for maximizing gross device consumption or communication performance [Sharma et al., 2010b, Ozel et al., 2011]. Similarly, RL approaches usually assume maximization of some single proxy objective such as duty cycle/sampling rate [Hsu et al., 2014, Shresthamali et al., 2017, Dias et al., 2016, Murad et al., 2019, Shresthamali et al., 2019, Xu et al., 2018, Xu et al., 2019] or

communication-based metric such as maximizing throughput/bitrate and minimizing packet loss/latency [Blasco et al., 2013, Ortiz et al., 2016, Ortiz Jimenez, 2019, Qiu et al., 2019, Kim et al., 2019, Van Huynh et al., 2019, Long and Büyüköztürk, 2020]. However, realistic implementations usually require optimizing over multiple objectives. In [Fei et al., 2016], the authors survey different analytic Multi-Objective Optimization (MOO) for WSNs.

However, not enough attention has been given to energy scheduling among multiple tasks w.r.t. different objectives in EHWSNs. This is an important topic, especially for modern sensor nodes such as the SenStick [Nakamura et al., 2017] which has been used for a variety of applications like human-activity recognition and indoor localization. Its operation involves activating its multiple sensors, processing the gathered data (using an ARM processor) and transmitting it via Bluetooth. Each of these tasks consume non-trivial amounts of power and this needs to considered when deciding the energy budget. Analytic methods deal with the dual objectives of ENO by recasting the problem with a single objective and a set of constraints which typically require predictive mechanisms [Kansal et al., 2007, Geissdoerfer et al., 2019]. Maximization of this objective is accomplished by methods such as convex optimization [Kansal et al., 2007], dynamic programming [Ozel et al., 2011] and control systems [Vigorito et al., 2007]. These solutions are not general because the constraints need to be set beforehand and they rely heavily on non-causal a priori information.

MOO using RL is especially difficult because the standard RL paradigm is based on learning from a *single* reward function. Solutions that explicitly distinguish between multiple objectives consolidate the rewards by some scalarization function [Hsu et al., 2014, Aoudia et al., 2018, Ferreira et al., 2018]. In [Fu et al., 2006], the authors represent a complex objective of transmission scheduling over a fading channel under deadline constraints using a single metric of reward per unit energy expended. Scalarization require some *a priori* application-specific knowledge about the emphasis on each objective. These specialized reward functions are difficult to design and also may lead to unexpected behavior due to reward hacking [OpenAI, 2020a, DeepMind, 2020, Everitt and Hutter, 2019]. Furthermore, the learning process has to be repeated

for each different preferences and tradeoffs cannot be made at runtime. Consequently, the user has to store a different policy for each unique preference, a solution which is quite unscalable. The drawbacks of reward scalarization is explored in [Vamplew et al., 2008] and the behavior of scalarized reward functions for ENO is explored in [Rioual et al., 2018]. As the number of objectives increase, the consequences of reward scalarization become even more severe. As it stands, EHWSNs would benefit from using a MORL approach for optimizing their operation because there is a possibility to learn from simpler multiple rewards and tradeoff between different objectives. We propose a MORL-based solution for energy scheduling in EHWSNs for our third and final contribution in Chapter 6. In addition to MOO using RL, our proposed solution requires less computation and training costs compared to existing methods and can deal with continuous state and actions spaces.

Interested readers can find a good overview of general MORL methods in [Liu et al., 2014]. MORL methods are generally available as single-policy methods and multi-policy methods. In single policy methods, the agent learns a policy that satisfies the preferences among the multiple objectives. These methods generally involve learning an action-value function that also takes into account the relative emphasis among the objectives as proposed in [Zeng et al., 2010, Ngai and Yung, 2011, Yang et al., 2019]. However, the proposed methods apply only for a discrete action spaces where performing an argmax search over the Q-values is feasible. This does not hold for continuous action spaces. Multiple-policy methods learn a set of policies in order to approximate the Pareto-frontier [Moffaert and Nowé, 2014, Shelton, 2001, Parisi et al., 2014, Pirotta et al., 2015]. However, we do not investigate this approach as this requires considerable computation resources over single policy methods.

While we acknowledge that computational resource availability is an important and pressing issue, it is not the focus of our work. However, having said that, we are mindful to use minimize the computational requirements of our learning algorithms as much as possible. Encouraging results have been reported in [Restuccia and Melodia, 2020] where the authors implement Deep RL to optimize the modulation scheme for software-defined radio in real-time low-power hardware. An orthogonal approach for

consolidating multiple objectives may be a game-theoretic/multi-agent approach as explored in [Sharma et al., 2019] and [Ortiz et al., 2017].

Chapter 3

Theory

3.1 Energy Neutrality

Regarding the operation of EHWSNs, we are basically interested in two things:

- **Energy Neutral Operation (ENO):** How to ensure that the energy consumed is always less than or equal to the amount of harvested energy?
- **Maximizing Performance:** What is the maximum performance level that can be achieved in a given harvesting environment while maintaining ENO?

We first develop the theory to ensure ENO and then proceed on to how we can maximize node performance while ensuring ENO.

Harvesting energy from the environment generally requires the following components:

1. **Harvesting Source:** The harvesting source scavenges energy from the ambient environment. The energy output of the source typically varies with time and environmental conditions and is outside the control of the designer.
2. **Load:** The load represents the energy consuming activity of the EHWSN. This may include sensing operations, data processing, control operations, actuation or data communication. The amount of energy consumed by the load varies with time. Most EHWSNs support multiple power consumption modes and

it is usually possible to have some control over the amount of energy being consumed by various methods such as sleeping/waking the device, Dynamic Voltage Frequency Scaling (DVFS) or duty cycling.

3. **Harvesting System:** This refers to the power management system that supports a variable load from a variable energy harvesting source. In our case, this includes the power manager along with the energy buffer (battery). The system ensures that the load's energy demands can be met even when the instantaneous power supply from the harvesting source may not match the load power consumption. Our objective is to learn a policy for ENO using RL.

3.1.1 Energy Neutral Operation (ENO)

Let us consider the case of a harvesting system equipped with an ideal battery with capacity B and initial charge B_0. We assume the battery capacity does not degrade with time, has no charging/discharging inefficiencies and does not leak energy. Let $P_s(t)$ be the power output from the energy source and $P_c(t)$ be the power consumption of the load EHWSN at any time t. The battery stores all the energy that is harvested. This energy can be used instantaneously by the load or at any other later time. For such a case, Equation 3.1 should be satisfied for all non-negative values of T.

$$\int_0^T P_c(t)dt \leq \int_0^T P_s(t)dt + B_0 \;\forall\; T \in [0, \infty) \tag{3.1}$$

Furthermore, since the battery has a *finite* capacity, B, we require the following Equation 3.2 to hold true if we do not want to waste any of the energy that has been harvested. This is not a *necessary* condition for ENO. However, if we want to achieve complete energy neutrality then Equation 3.2 has to be true.

$$B_0 + \int_0^T P_s(t)dt - \int_0^T P_c(t)dt \leq B \;\forall\; T \in [0, \infty) \tag{3.2}$$

3.1.2 Source and Load Models

We now define a function to model variable energy sources. The first parameter in this model is the average rate at which energy is provided by the source. We also need parameters to characterize the variability inherent in the energy production profile. We thus define a model as follows:

Definition 3.1.1. $(\rho, \sigma_1, \sigma_2)Function$: A nonnegative, continuous and bounded function $P(t)$ is said to be $(\rho, \sigma_1, \sigma_2)$ function if, and only if, for any value of finite positive real numbers τ and T, the following are satisfied:

$$\int_{\tau}^{\tau+T} P(t)dt \leq \rho T + \sigma_1 \tag{3.3}$$

$$\int_{\tau}^{\tau+T} P(t)dt \geq \rho T - \sigma_2 \tag{3.4}$$

Definition 3.1.1 can be used to model variable energy sources as well as loads. The unit of ρ is power (e.g., Watts) and that of σ_1 and σ_2 is energy (e.g., Joules).

A variable load can be modeled as a special case of Definition 3.1.1. Let us assume the average power consumption of the load is ρ_2 with an upper bound of σ_3. The lower bound is zero which means that the load can have zero consumption mode. This gives us Equation 3.5 and Equation 3.6 as follows:

$$\int_{T} P_c(t)dt \leq \rho_2 T + \sigma_3 \tag{3.5}$$

$$\int_{T} P_c(t)dt \geq 0 \tag{3.6}$$

3.1.3 Battery Capacity

Now that we have modeled the energy sources and loads, we revisit Equation 3.1. Consider the worst-case scenario when the source is producing the least amount energy and the load is consuming the highest possible amount of energy. In this case, Equation 3.1 can be written as

$$\max\left\{\int_T P_c(t)dt\right\} \leq \min\left\{\int_T P_s(t)dt\right\} + B_0 \tag{3.7}$$
$$\Rightarrow \rho_2 T + \sigma_3 \leq \rho_1 T - \sigma_2 + B_0$$

We ensure ENO by requiring Equation 3.7 to be valid for all time $T \geq 0$. Setting $T = 0$, we get

$$\sigma_2 + \sigma_3 \leq B_0 \tag{3.8}$$

Equation 3.8 gives us the least amount of *initial* storage in the battery required to ensure ENO. This means that the battery must have a large enough initial charge to ensure node operation even when the node is consuming the highest possible energy while the amount of energy harvested is at its lowest.

Next, taking the limit $T \to \infty$, we get

$$\rho_2 \leq \rho_1 \tag{3.9}$$

Equation 3.9 tells us that the average power production of the source should exceed or at least be equal to the average power consumption of the load.

To determine the battery capacity, we revisit Equation 3.2 and introduce our model parameters. We then consider the other extreme worst-case scenario when the load is consuming the least amount of energy (i.e., zero in this case) while the source is producing the most.

$$B_0 + \max\left\{\int_T P_s(t)dt\right\} - \min\left\{\int_T P_c(t)dt\right\} \leq B \tag{3.10}$$
$$\Rightarrow B_0 + (\rho_1 T + \sigma_1) - 0 \leq B$$

Setting $T = 0$, we get

$$B_0 + \sigma_1 \leq B \tag{3.11}$$

Using Equation 3.8, the constraints on the required battery size is given by

$$\sigma_1 + \sigma_2 + \sigma_3 \leq B \tag{3.12}$$

Thus, if we want to ensure energy neutrality, the battery should be large enough such that it can store the required initial charge (Equation 3.11) and have enough capacity to store all the excess energy that is harvested but cannot be consumed immediately (Equation 3.12).

3.1.4 Perpetual Operation

From the above definitions and results, we can now state the following theorem.

Theorem 3.1.1. *Perpetual Energy Neutral Operation: Consider an energy harvesting system with an ideal battery in which the energy production profile is a $(\rho_1, \sigma_1, \sigma_2)$ function and the load consumption profile is a (ρ_2, σ_3) function. The following conditions are sufficient for perpetual (energy neutral) operation.*

$$\rho_2 \leq \rho_1 \tag{3.13}$$

$$\sigma_2 + \sigma_3 \leq B_0 \tag{3.14}$$

$$B_0 \leq B \tag{3.15}$$

where B is the capacity of the battery and B_0 is the initial energy stored in the battery.

Theorem 3.1.1 gives us the condition for eliminating downtimes but doesn't prevent overflows. It only assures that as long as the given conditions hold, the node will function perpetually. This does not guarantee that none of the energy will go to waste. We need to ensure that all harvested energy is completely used up for true energy neutrality.

If we assume that the load has a constant power consumption profile i.e., $\sigma_3 = 0$, then we can state Theorem 3.1.2 [Kansal et al., 2004] using Equation 3.12.

Theorem 3.1.2. *Energy Neutral Operation: If a device is powered by a*

- $(\rho_1, \sigma_1, \sigma_2)$ *source*
- *operates at constant power* ρ_1*,*
- *has an ideal battery with initial charge of* $B_0 \geq \sigma_2$
- *and maximum capacity* $B \geq \sigma_1 + \sigma_2$

then the device utilizes the energy source fully and can operate forever.

We are now at a position such that given an energy source that can be characterized as a $(\rho_1, \sigma_1, \sigma_2)$ source, we can calculate the

- minimum battery size $(\sigma_1 + \sigma_2)$,
- minimum initial battery charge (σ_2) and
- the load power consumption rate (ρ_1)

to ensure ENO. However, in real world applications, the device power consumption rate (ρ_2) cannot be expected to match exactly with ρ_1. We thus need to adapt the power of the load accordingly to maintain energy neutrality.

3.1.5 Duty Cycling

In our research, we use duty-cycling to adjust the load power. We assume that the utility of the load (sensor node) to the user, $U(D)$, when operated at a duty cycle D is characterized as follows:

$$U(D) = 0, \qquad if\ D < D_{min} \tag{3.16}$$

$$U(D) = k_1 + k_2 D \qquad if\ D_{min} \leq D \leq D_{max} \tag{3.17}$$

$$U(D) = k_3 \qquad if\ D > D_{max} \tag{3.18}$$

The above characterization is general and implies that a duty cycle below D_{min} is useless to the user. On the other hand, increasing the duty cycle above D_{max} has no additional utility to the user either. If P_{max} is the maximum power consumption of the load, we can operate the load at constant power ρ_2 by fixing the duty cycle D, $0 \leq D \leq 1$, such that

$$\rho_2 = P_{max} \times D \tag{3.19}$$

3.1.6 Optimal Policy

We now proceed to calculate the optimal power-usage policy that is possible for a given energy generation profile. To do this, we assume that we have *complete* knowledge of the energy availability, including future knowledge.

Suppose the time is discretized into slots of duration ΔT and the duty cycle adaptation is carried out over a window (or episode) of N_E slots. We define the following discretized versions of energy profile variables, with index i ranging over $\{1, ..., N_E\}$:

- $P_s(i)$, the power input from harvesting source in slot i. We assume that this power is constant over the slot duration.
- P_{max}, the maximum power consumption of the load. We can decrease the load power consumption by changing the load duty cycle.
- $D(i)$, the duty cycle used in slot i. We would like to calculate the value of $D(i)$ for all i to ensure energy neutrality.
- $B(i)$, the residual battery at the *beginning* of slot i. The battery energy left after the last slot in the window is $B(N_E + 1)$.

With the above definitions, the residual battery at the beginning of a time slot i can be expressed as:

$$B(i+1) = B(i) + \Delta T P_s(i) - \Delta T D(i) P_{max} \tag{3.20}$$

For energy neutral operation, it is sufficient that the battery at the end of the window of N_E, $B(N_E+1)$, be greater than or equal to the initial battery level, $B(1)$. However, if we want to ensure energy neutrality, then the battery at the end of the window has to be *exactly equal* to the initial battery level.

In other words, at the end of the window, the initial and final battery levels must be equal. This ensures that all of the energy harvested was used up and the battery suffered no net energy change.

We now express the calculation of optimal duty cycles for energy neutrality as the following optimization problem:

$$\max \sum_{i=1}^{N_E} D(i) \tag{3.21}$$

with the constraints,

$$B(i+1) = B(i) + \Delta T P_s(i) - \Delta T D(i) P_{max} \ \forall \ i \in \{1, ..., N_E\} \tag{3.22}$$

$$B(1) = B_0 \tag{3.23}$$

$$B(N_E+1) = B_0 \tag{3.24}$$

$$B(i) \leq B \tag{3.25}$$

$$D(i) \geq D_{min} \ \forall \ i \in \{1, ..., N_E\} \tag{3.26}$$

$$D(i) \leq D_{max} \ \forall \ i \in \{1, ..., N_E\} \tag{3.27}$$

where B_0 is the starting residual battery energy.

The solution to the above problem gives us the values of $D(i) \ \forall \ i \in \{1, ..., N_E\}$. This optimization problem can be solved using linear programming techniques. It is important to note that the solution may not be unique i.e., there may be multiple strategies that can achieve energy neutrality. This is because we have considered here the special case in which the battery is ideal. The general case with battery inefficiencies is described in [Kansal et al., 2007].

We now know how to arrive at the optimal policy using linear optimization provided we have access to perfect non-causal information about energy availability.

However, this is not a realistic solution. Instead, we would like to use a action-perception-learning approach to arrive at the optimal policy as discussed in Chapter 1. Reinforcement Learning (RL) fits neatly with this approach. We now proceed to a brief introduction of RL and its related theory.

3.2 Reinforcement Learning (RL)

Reinforcement learning is a machine learning technique suitable for sequential decision making tasks. The learning agent experiences various situations (states) and has to decide what actions to execute. On execution of an action, it receives feedback in the form of a scalar reward signal. The agent's objective is to choose actions such that the cumulative reward signal is maximized. In a general case, the objective is to maximize the *discounted cumulative* reward.

In RL, the agent is not told what actions to take. This is left to the agent to learn via experience i.e., by trying out different actions and figuring out which is the most rewarding. In the general case, the actions chosen not only affect the amount of reward received but also the consequent states that are experienced by the agent.

The learning process requires the agent to search its state space and try out different actions. This is called *exploration*. The agent memorizes which states and actions were profitable (and to what degree) and which were not. Using this knowledge gained through experience, the agent chooses more wisely when it encounters a similar situation. This is called *exploitation* (of learned knowledge).

The trade-off between exploration and exploitation is one of the major challenges in RL. It is necessary for the agent to repeat the most effective actions as often as possible to accumulate the most reward. However, it is also equally important that the agent explore various states and actions before it can decide upon the *most effective* one. The agent has to exploit its learned knowledge to get the maximum reward but it also has to explore to figure out better action selections in the future. The agent cannot hope to succeed if it exclusively follows either exploration or exploitation policies. Instead, it must try different actions and progressively favor those that are

the most rewarding.

As mentioned before, the agent is not told what actions to take to achieve the maximum cumulative reward. This is in contrast to supervised learning methods. Supervised learning methods require a supervisor that gives learning agents the *correct* answer for each training input. The feedback in this case is *instructive* - the supervisor tells what is the correct answer. In contrast, an RL agent receives feedback in terms of rewards. It is up to the learning agent to evaluate this reward and decide whether the corresponding action was a good one or not i.e., the feedback is *evaluative*.

Supervised learning methods are quite ill suited for interactive decision-making problems. This is because it is quite impractical to assemble a training set of all the examples of desired behaviors that are both correct and representative of all of the agent's states.

3.2.1 Markov Decision Process (MDP)

RL is best suited for sequential decision making tasks. Generally, the reward due to a particular actions may be delayed temporally. Such RL tasks with delayed reward signals are well modeled using a MDP. Here, we will only be concerned with situations where the state and action spaces are finite. We also assume that the agent-environment interactions happen in discrete time steps.

The MDP for a general RL system shown in Figure 3-1 is defined by

- a set of states (state space), S
- a set of actions (action space), A
- a reward function $R : S \times A \rightarrow R$. This function $R(s, a)$ gives the agent a scalar reward r for executing action a in state s.
- a state transition function $T : S \times A \rightarrow \Pi(S)$, where a member of $\Pi(S)$ is a probability distribution over the set S. The transition function maps states to probabilities. $T(s, a, s')$ is the probability that agent transitions from state s to s' as a result of action a.

- a policy $\pi = \{(s, a)|a \in A, s \in S\}$. This means that the policy π tells us which action the agent will choose as a function of its state. This is denoted by $\pi(s) = a$.

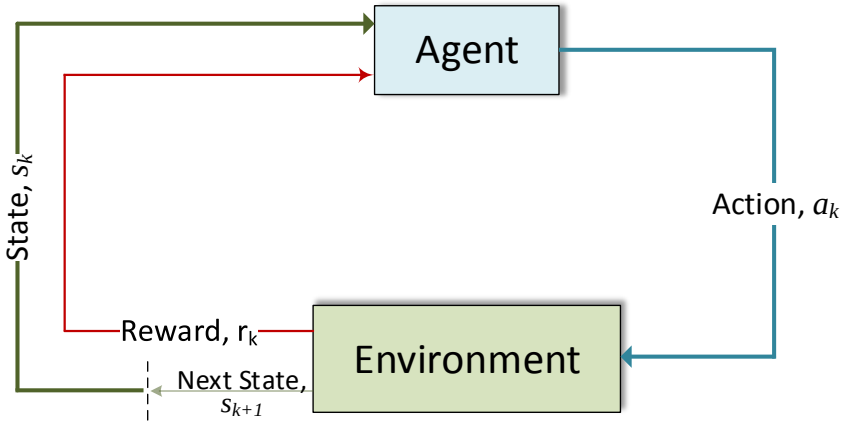

Figure 3-1: A general RL system consist of an agent that interacts with its environment in the form of actions. The environment reacts to these actions in the form of reward signals.

Each agent-environment interaction incurs a timestep. At timestep $t = k$, the agent determines it is in state $s_k \in S$. It does so by evaluating the various input from the environment. The agent takes an action, $a_k \in A$, according to some policy π. The environment reacts to this action by changing the agent's state to a new state $s_{k+1} \in S$ and rewarding the agent with some scalar reward k_t. An *episode* E, consists of such N_T consecutive timesteps where $t = T$ is the terminal timestep. It is possible that some agent-environment interactions will never terminate but continue on endlessly. We do not discuss such interaction in this dissertation. We concern ourselves only with episodic tasks i.e., an episode of agent-environment interaction terminates after a finite number of timesteps.

The objective of the agent is to find a policy that maximizes at each time step the expected discounted sum of future reward. We discount future rewards to express the fact that immediate present rewards have more weight than future probable rewards. An optimal policy, π^*, maximizes the cumulative reward for all state-action pairs.

The RL *task* is to find the optimal policy. There are various ways of going about this. Generally, these methods may require the agent to create a model of the environment and plan its policy using that model. This model typically tries to approximate the state-transition function. In our case, we consider a *model-free* RL paradigm where the agent determines its policy without bothering with the transition probabilities and learning an explicit model of the environment.

3.2.2 Q-Values

Let us assume our agent is acting according to some policy π. The cumulative discounted *return* at a time t is

$$G_t = \sum_{t'=t}^{T} \gamma^{t'-t} r_{t'} \tag{3.28}$$

where T is the timestep of episode termination and γ, $0 < \gamma < 1$, is the discounting factor.

To give a measure of the the *goodness* of a particular action, a, according to the policy π, when the agent is in some state s, we assign each state-action pair a Q-value [Sutton et al., 1992]. The Q-value $Q^\pi(s, a)$ is the *expected* sum of discounted rewards (or return) when executing action a from state s and following policy π thereafter. Mathematically it is expressed as,

$$Q^\pi(s, a) = \mathbb{E}[\, G_t | s_t = s, a_t = a, \pi] \tag{3.29}$$

It can also be computed recursively by using the Bellman equation:

$$Q^\pi(s_t, a_t) = \mathbb{E}_\pi[R(s_t, a_t) + \gamma Q(s_{t+1}, \pi(s_{t+1}))] \tag{3.30}$$

The Q-values for the optimal policy π^* is denoted by Q^*. The optimal action-value function is therefore given by

$$Q^*(s, a) = \operatorname*{argmax}_\pi Q^\pi(s, a) \tag{3.31}$$

Once Q^* is known, we can then determine the optimal action. For any state s, the action a^* that maximizes $Q^*(s,a)$ is the optimal action or the *greedy* action. When the actions space is discrete and limited to a small number of actions, the optimal policy is a greedy policy given by

$$\pi^*(a|s) = a^* = \underset{a}{\operatorname{argmax}}\, Q^*(s,a) \tag{3.32}$$

$Q^*(s,a)$ can also be recursively computed using the Bellman equation,

$$Q^*(s,a) = \mathbb{E}\big[r + \gamma \max_{a'} Q^*(s',a')\big], \tag{3.33}$$

where (s',a') are the next state-action pair and the Bellman target is $r+\gamma \max_{a'} Q^*(s',a')$.

For a general stochastic policy π, the *value of a state* i.e., the measure of how good a particular state s is is given by:

$$V(s) = \mathbb{E}_{a\sim\pi(a|s)} Q^\pi(s,a) \tag{3.34}$$

The advantage function $A^\pi(s,a)$ gives a measure of how "better" an action is compared to other alternative actions for policy π.

$$A^\pi(s,a) = Q^\pi(s,a) - V^\pi(s) \tag{3.35}$$

When we are dealing with a discrete state-action space, it is tractable to record the Q-values in a tabular form. In such cases, finding a^* is trivial once Q^* is known and the actions space consists of only a few discrete actions. The real challenge here is to determine (learn) the values of Q^* by using an RL algorithm that balances exploration and exploitation. There are various algorithms that accomplish this. For the tabular case, we will focus on the SARSA(λ) which is described in Section 3.2.3. For continuous state spaces with a discrete action space, we can replace the Q-table with a Q-function. The Q-function can be learned using function approximation [Sutton et al., 1992, Mnih et al., 2013, Mnih et al., 2015, Silver et al., 2016]. We will

discuss this approach in Section 3.2.7. When the action space is continuous, we use policy gradient [Munos, 2006, Silver et al., 2014] methods. DDPG is one such method and is described in Section 3.2.8.

3.2.3 SARSA

SARSA stands for State-Action-Reward-State-Action. SARSA is a Temporal Difference (TD) learning algorithm Algorithm 1. Temporal difference means that the difference (error) in $Q(s, a)$ at different times (e.g., before action a is executed versus after the action is executed) is used to update $Q(s, a)$. This difference in Q-values at different timesteps is known as the TD-error. With enough time and experience, SARSA can converge to the true values by minimizing TD-errors directly from raw experience without requiring a model of the environment.

During SARSA, the agent maintains an estimate of the Q-values. These estimates are updated in part from other learned estimates without waiting for the final outcome i.e., via bootstrapping. SARSA is an *on-policy* TD method (as opposed to *off-policy* algorithms like Q-learning). This means that the SARSA learns the Q-values Q^π, for policy π using the experience gathered when following policy π. In other words, the *learned* Q-values Q^π, correspond to the *behavioral* policy π. This is different from off-policy methods where the Q-values Q^ζ, for some policy ζ, are learned using the experiences gathered from an arbitrarily different policy μ.

The SARSA algorithm requires that the agent maintain a table, $Q^\pi(s, a)$ with Q-values for all state-action pairs. Initially, the Q-table can be arbitrarily initialized. The agent starts out in state s_k, takes action a_k according to some policy π. As a result, it receives a reward r_k and transitions to another state s_{k+1}. The agent then considers taking the next action a_{k+1} according to the policy π.

Now that we have r_k and the next state-action pair (s_{k+1}, a_{k+1}), we can build a better estimate of $Q^\pi(s_k, a_k)$ using Equation (3.30). We can use this new estimate $r_k + \gamma Q^\pi(s_{k+1}, a_{k+1})$ to update $Q^\pi(s_k, a_k)$ as follows:

$$Q^{\pi}(s_k, a_k) \leftarrow Q^{\pi}(s_k, a_k) + \alpha \left[r_k + \gamma Q^{\pi}(s_{k+1}, a_{k+1}) - Q^{\pi}(s_k, a_k) \right] \tag{3.36}$$

The constant, α, is the learning rate and it determines how much of the Q-values are modified in each step. Equation (3.36) can be rearranged to give:

$$Q^{\pi}(s_k, a_k) \leftarrow (1 - \alpha) Q^{\pi}(s_k, a_k) + \alpha \left[r_k + \gamma Q^{\pi}(s_{k+1}, a_{k+1}) \right] \tag{3.37}$$

Algorithm 1 shows the SARSA learning process. Thus, we continually estimate Q^{π} for the behavior policy π. Updating Q^{π} brings π closer to convergence [Sutton et al., 1992]. SARSA is guaranteed to converge to the correct Q-values as long as all state-action pairs are visited an infinite number of times.

Algorithm 1: SARSA algorithm

```
1  Initialize Q^π(s, a) arbitrarily
2  Observe current state s_k
3  for each episode do
4      repeat
5          for each timestep do
6              Choose and execute action a_k according to policy π
7              Receive reward r_k
8              Observe new state s_{k+1}
9              Determine the next action, a_{k+1} = π(s_{k+1})
                 Q^π(s_k, a_k) ← (1 − α)Q^π(s_k, a_k) + α [r_k + γQ^π(s_{k+1}, a_{k+1})]
10             s_k = s_{k+1}
11         end for
12     until end of episode;
13 end for
```

3.2.4 ϵ-greedy policy

Unlike supervised learning methods, RL methods are not divided into training and testing modes. The closest we can come to this differentiation is by trading off between exploration and exploitation modes. When the agent is just starting to learn, we might want the agent to explore as much as possible about the environment. We call

this the exploration mode. Once it has *sufficiently* explored, we can then exploit this learned knowledge to then concentrate our efforts to accumulate as much reward as possible. This is called the exploitation mode. One way of balancing exploration and exploitation is by using a ϵ-greedy policy.

A greedy policy is that policy which always chooses the greedy action i.e., the action that maximizes the Q-value for a particular state. However, acting greedily before convergence of $Q(s, a)$ to true values may lead to sub-optimal policies because the agent would not have had the opportunity to sample state-actions pairs that might have led to higher returns. Premature convergence to incorrect Q-values leads to sub-optimal returns and fragile policies.

On the other hand, an ϵ-greedy policy means that the actions are chosen greedily most of the time, but with probability ϵ, a random exploratory action is selected. Thus setting a high value of ϵ means the agent is in exploration mode whereas setting a low value of ϵ means the agent is in exploitation mode.

During SARSA learning, it is advisable to start with high values of ϵ and gradually decrease its value with more experience. Ultimately, once the Q-values have converged, we can set ϵ to zero and pursue a purely exploitative greedy policy.

We can also nudge the agent towards exploring remote states and actions in the early stages of learning by optimistically initializing the Q-table [Sutton et al., 1992]. This is done by assigning all state-action pairs high initial Q-values [Sutton et al., 1992].

3.2.5 Eligibility Traces

In our discussions, we have assumed that the agent receives rewards after every timestep. However, it is entirely possible that during an episode of agent-environment interaction, the reward is awarded only at the end of the episode i.e., intermediate rewards are zero. In such cases, the TD learning may take a long time to converge because the rewards are sparse and delayed.

It is possible to increase the learning efficiency by identifying which state-action pairs contributed in securing the reward at the end of the episode. To accomplish

this, we introduce a memory variable for each state-action pair called the *eligibility trace*. The eligibility trace for a state-action pair at timestep $t = k$ is denoted by $e_k(s, a) \in \mathbb{R}_{\geq 0}$. The eligibility trace at the end of an episode $e_T(s, a)$, indicates how much the state-action pair (s, a) contributed to the acquisition of the reward at the end of the said episode.

The mechanism for eligibility trace is as follows. The agent maintains a table of eligibility traces, called the e-table, for all state-action pairs. The e-table is very much like the Q-table but contains $e(s, a)$ instead of the Q-values, $Q(s, a)$. The eligibility traces are initially initialized to zero.

During each timestep, the agent decays all the entries in the e-table by a factor of $\gamma\lambda$ where γ is the discount factor and λ is called the *trace-decay parameter*. However, the eligibility trace for the state-action pair visited on that timestep is also incremented by 1.

Thus, assuming that our agent is in state s_k and executes action a_k during timestep k. Then,

$$e_k(s, a) = \begin{cases} \gamma\lambda e_{k-1}(s, a) & \text{if } (s, a) \neq (s_k, a_k) \\ \gamma\lambda e_{k-1}(s, a) + 1 & \text{if } (s, a) = (s_k, a_k) \end{cases} \tag{3.38}$$

for all (s, a), where $0 < \lambda < 1$.

For the sake of example, Figure 3-2 shows the eligibility trace for the state-action pairs (s_1, a_1) and (s_2, a_2) in an arbitrary episode. At the end of the episode, some reward, R was received. We can observe that (s_1, a_1) was visited quite frequently in the beginning of the episode but not at all during the end of the episode. On the other hand, (s_2, a_2) was visited very often by the agent during the later parts of the episode. It is reasonable to say (s_2, a_2) had more influence in securing reward R as compared to (s_1, a_1). This *influence* is reflected on the values of $e(s_1, a_1)$ and $e(s_2, a_2)$ as shown in the diagram. Thus the temporal difference error for (s_2, a_2) should be more heavily weighted as compared to (s_1, a_1). This is the basic concept of using eligibility traces to update Q-values. The use of eligibility traces allows us to propagate the TD-error backwards to all the state-action pairs that contributed to

it with most recent state-action pairs having exponentially higher weights than the ones further in the past.

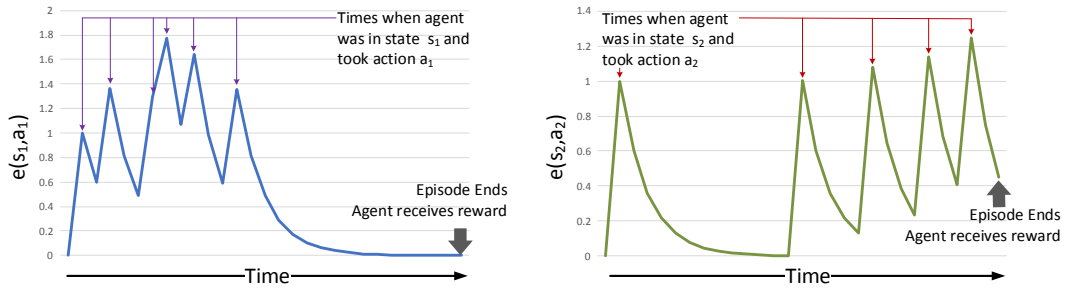

Figure 3-2: Eligibility traces for (s_1, a_1) and (s_2, a_2)

Setting a low λ means that the TD-error contributes less in the update of Q-values of temporally distant state-action pairs i.e., earlier state-action pairs are given less credit than recent ones. A high λ means that the update of state-action pairs that are temporally distant are also significantly affected by the TD error.

We can use eligibility traces in the previous SARSA algorithm Algorithm 1 to achieve better performance. The corresponding algorithm is called SARSA(λ). The SARSA(λ) algorithm using ϵ-greedy policy is shown listed below in Algorithm 2.

3.2.6 Q-Learning

As explained before, SARSA is an *on-policy* RL algorithm where the Q-values corresponding to the behavioral policy are learned. It is also possible to learn the Q-values corresponding to a policy using experiences gathered using a different behavioral policy. Q-learning [Watkins and Dayan, 1992, Sutton et al., 1992] is one such off-policy RL method. During Q-learning, the Q-values corresponding to the *optimal* policy π^* are learned from experiences gathered from an arbitrary policy. In some cases, it is possible to learn the optimal policy by simply using a completely random policy during Q-learning.

The Q-Learning algorithm is very similar to SARSA. For the tabular case, the agent maintains an arbitrarily initialized table, $Q(s, a)$ with Q-values for all state-

Algorithm 2: SARSA(λ) algorithm

```
Initialize Q(s,a) = qinit and e(s,a) = 0 for all s,a
for each episode do
    Initialize s,a
    for each step of the episode do
        Take action a, observe reward r and next state s'
        Choose next action a' from Q using ε-greedy policy
        δ ← r + γQ(s',a') − Q(s,a)
        e(s,a) ← e(s,a) + 1
        for all (s,a) do
            Q(s,a) ← Q(s,a) + αδe(s,a)
            e(s,a) = γλe(s,a)
        end for
        s ← s'
        a ← a'
    end for
end for
```

action pairs. At each transition from timestep k to $k+1$, the agent observes s_k, a_k, r_k and s_{k+1}. Based on these, the Q-value estimate is updated at timestep $k+1$ according to the following formula.

$$Q(s_k, a_k) \leftarrow (1-\alpha)Q(s_k, a_k) + \alpha \left[r_k + \gamma \max_a Q(s_{k+1}, a) \right] \tag{3.39}$$

Equation 3.39 show that the value of $Q(s,a)$ is updated *assuming* that the next action will be the *greedy* action. Algorithm 3 shows the Q-learning process that uses a behavioral policy π. Due to the fact that Q-learning is an offline learning method, the choice of π is inconsequential with respect to the final Q^* values. It has been shown that the Q-values ultimately converge to the optimal Q-values if all actions continue to be tried from all states [Sutton et al., 1992, Watkins, 1989]. However, in reality, using a suitable π can hasten the convergence to Q^*.

3.2.7 Deep Q-Networks (DQN)

For discrete states and actions, Q(s,a) can be stored in a tabular form where each entry in the table is the Q-value of a particular state and action. This method of

Algorithm 3: Q-learning algorithm

Initialize $Q(s, a)$ to arbitrary initial values
Observe current state s_k
for *each episode* **do**
  repeat
    for *each timestep* **do**
      Choose and execute action a_k according to policy π
      Receive reward r_k
      Observe new state s_{k+1}
      $Q(s_k, a_k) \leftarrow (1 - \alpha)Q(s_k, a_k) + \alpha \left[r_k + \gamma \max_a Q(s_{k+1}, a)\right]$
      $s_k = s_{k+1}$
    end for
  until *end of episode*;
end for

storing Q-values is extremely efficient as it requires only some amount of memory to store the table and a lookup operation to access the Q-values. This makes it a very attractive solution for IoT nodes that are constrained in computational resources. However using Q-tables is limited only to discrete state-action spaces. Furthermore, it is particularly difficult to learn the Q-values for high dimensional states and actions due to the curse of dimensionality. By using function approximation methods, we can extend Q-learning for continuous states. In this section, we focus on using NN for function approximation - also known as DQNs. Specifically we use a variant of DQN called Double DQN [Van Hasselt et al., 2016] with dueling architecture [Wang et al., 2015b].

DQNs use NNs to approximate the action-value function $Q(s, a) \sim Q(s, a; \theta)$ for a given state-action pair. The parameter θ of DQN constitute the weights and biases for the NN and are optimized to approximate the Bellman equation. Function approximation using neural networks for Q-learning is very sample inefficient i.e., it require lots of experiences before it can learn a good policy. DQNs overcome this by using *experience replay* [Lin, 1993, Mnih et al., 2013]. Experience replay simply means that the experiences are stored in a memory buffer so that they can be reused again and again to increase sample efficiency. During learning, the memories are randomly chosen from the buffer which decreases the correlation between them and

therefore improves the NN learning performance.

On the i-th iteration, the action-value function, $Q(s, a; \theta_i)$, is approximated by a DQN with parameters θ_i. The agent stores its experiences $m_t = (s_t, a_t, r_t, s_{t+1})$ at each timestep t in a memory pool $\mathcal{M} = \{m_1, m_2....m_t\}$. During learning, random minibatches of experiences $(s, a, r, s') \sim \mathcal{Z}(\mathcal{M})$ are selected for updating θ through gradient descent methods using the following loss function.

$$L_i(\theta_i) = \mathbb{E}_{(s,a,r,s')\sim\mathcal{Z}(\mathcal{M})}\left[\left(r + \gamma \max_a Q(s', a; \theta_i) - Q(s, a; \theta_i)\right)^2\right] \tag{3.40}$$

Target Networks

The loss function in Equation (3.40) updates θ so that the difference between $Q(s, a : \theta_i)$ and its target $r + \gamma \max_a Q(s', a; \theta_i)$ is minimized. However, since Q-function and the target value both are parameterized by θ_i, the learning process is very susceptible to oscillations, divergence and long learning times. To overcome this, DQNs employ two different neural networks, $Q(s, a; \theta)$ and $Q(s, a; \theta^-)$ for added stability [Mnih et al., 2015] - one containing the most recently updated set of parameters θ and the other containing an older(frozen) version of the parameters θ^-. θ is updated at every learning step. θ^- receives a copy of the most recently updated parameters θ every N_X learning steps. Thus, the loss function now becomes:

$$L_i(\theta_i) = \mathbb{E}_{(s,a,r,s')\sim\mathcal{Z}(\mathcal{M})}\left[\left(r + \gamma \max_a Q(s', a; \theta_i^-) - Q(s, a; \theta_i)\right)^2\right] \tag{3.41}$$

In Equation (3.41), the target values are calculated using a different network parameterized by θ_i^- instead of θ_i like in Equation (3.40). This method has been shown to stabilize the learning process [Mnih et al., 2015, Hessel et al., 2018].

Double DQN

DQNs also suffer from overestimation bias [Hasselt, 2010, Thrun and Schwartz, 1993] due to the max operation during greedy action selection. During the beginning of training, the Q-values estimated by the DQN are generally erroneous due to bad initialization and lack of exploration. By taking the max over these faulty Q-values, the network is prone to overestimating its actions and adopting overoptimistic Q-value estimates. This is because the same network (parameterized by θ^-) is used for evaluating the target Q-value *and* choosing the greedy action. In [Van Hasselt et al., 2016], the authors propose Double DQN, a method to decouple the action selection and action evaluation using Double Q-Learning [Hasselt, 2010]. This method updates the NN parameters using the following loss function.

$$L_i(\theta_i) = \mathbb{E}_{(s,a,r,s')\sim\mathcal{Z}(\mathcal{M})}\left[\left(r + \gamma Q(s', \underset{a}{\operatorname{argmax}}\, Q(s', a; \theta_i); \theta_i^-) - Q(s, a; \theta_i)\right)^2\right] \quad (3.42)$$

By using the loss function in Equation (3.42), the greedy action selection is performed using Q-values calculated by the NN with parameters θ while the target Q-values are computed by using the frozen NN with parameters θ^-.

Dueling Architecture

Both vanilla DQN and Double DQN use NN to estimate the Q-values of different actions for a given state. An alternative to directly computing Q-values is to compute the value of the *state* $V(s)$, and the *advantage* $A(s)$, of the different actions separately. If we refer back to Equation (3.34) and Equation (3.35), we can express the Q-value as:

$$Q^\pi(s, a) = V^\pi(s) + A^\pi(s, a) \quad (3.43)$$

Dueling architectures [Wang et al., 2015b] were introduced so as to decouple the computation of the value of the states and the advantage of the actions. This is

achieved by splitting the NN architecture into two separate streams (Figure 3-3). Each stream consists of fully connected layers which explicitly represent either $V(s)$-values or $A(s)$-values.

By doing so, dueling networks learn which *states* are valuable during each update to the value stream V. This enables them to learn much faster than vanilla single-stream methods. This is because, for single-stream DQN, an update only affects the value estimation for one particular action. Using dueling architectures allows the agent to optimize its policy in a state space where it is sometimes unnecessary to estimate the value of each action choice. Additionally, using advantage estimation makes this NN more robust against small noises in Q-value estimates which may induce jitters during action selection. By using dueling architectures with Double DQN loss functions, we can achieve much faster and stable learning compared to vanilla DQN methods.

3.2.8 Deep Deterministic Policy Gradient (DDPG)

The tabular SARSA algorithm required discrete state and actions spaces. DQNs on the other hand were able to generalize over continuous state spaces making them much more powerful and general. However, DQNs still require the action space to be discrete and small. If the action space were very large (or continuous), the max operation over the different Q-values would not be possible. We thus require a solution that deals with continuous state spaces *and* continuous action spaces.

DDPG is an off-policy actor-critic formulation that can accommodate continuous state-action spaces. Off-policy points to the fact that the algorithm is able to learn the Q-function for a policy π although its training examples maybe obtained from a (slightly) different policy. Since we are considering continuous state spaces like in the case of DQN, we approximate the Q-values with a neural function approximator $Q_\theta(s, a)$ parameterized by θ. This action-value function is referred to as the *critic*.

In the discrete action case, given a Q-function $Q^*(s, a)$, the optimal policy π^* was

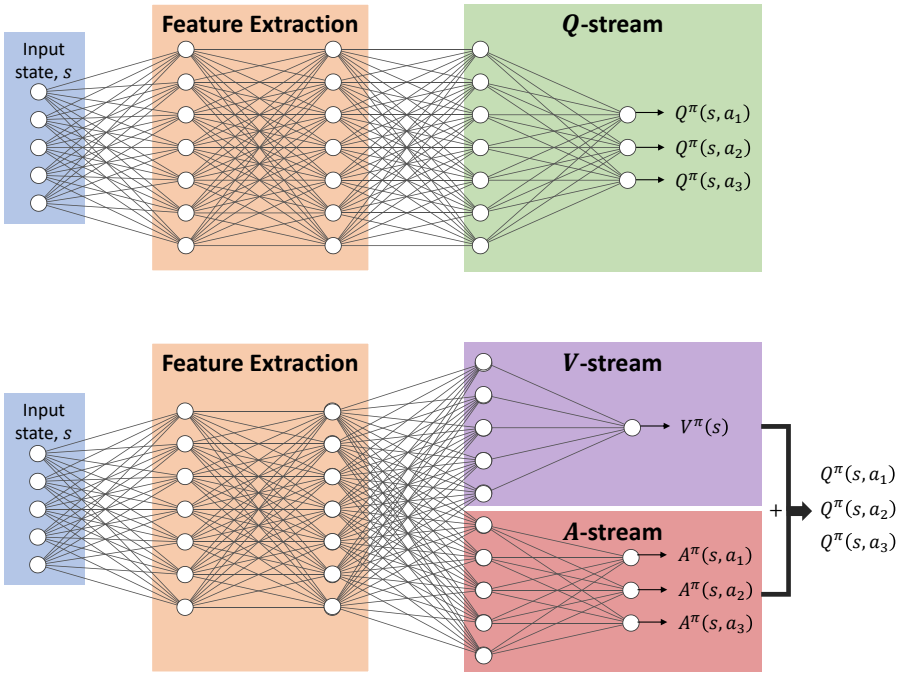

Figure 3-3: NN architectures for vanilla DQN (top) and Dueling Architectures (bottom). The state space has 5 parameters and there are three possible actions. Both NNs output the Q-values for all actions $Q(s, a)$. However in the dueling NN (bottom), the final layers are split into the V-stream to predict the value of state s, and the A-stream to predict the advantage of each of the actions. The outputs of the streams are summed to give the final Q-values

quite trivial to determine. We simply chose the action with the maximum Q-value.

$$\pi^*(a|s) = \underset{a}{\text{argmax}}(Q^*(s, a)) \tag{3.44}$$

However, for continuous actions, we cannot argmax over the Q-values. So instead, we represent the policy by a NN function approximator $\pi_\phi(s)$ with parameters ϕ. This policy function is referred to as the *actor*. $\pi_\phi(s)$ outputs the action (in the form of a continuous value) for a given state s.

Thus, in the DDPG actor-critic system [Sutton and Barto, 2018], the critic evaluates the Q-value of a state-action pair whereas the actor outputs a deterministic action for a given state. To encourage exploration, some random noise is added

to the actor's output during the learning phase. Similar to DQN, we also maintain time-delayed versions of the actor and critic referred to as *target* networks[Mnih et al., 2015], $Q_{\theta^-}(s,a)$ and $\pi_{\phi^-}(s)$, to increase learning efficiency and stability. The parameters of the target networks are updated by polyak averaging with the parameters of their main counterparts.

During training, the agent collects experience tuples and stores it in an experience replay buffer similar to DQN. At each training step, a random minibatch $\mathbb{B}$ of experiences are selected and a gradient *descent* is performed to minimize the loss functions in Equation (3.45a) and Equation (3.45b).

$$L(\theta, \mathbb{B}) = \mathbb{E}_{\mathbb{B}}\left[Q_\theta(s,a) - (r + \gamma Q_{\theta^-}(s', \pi_{\phi^-}(s'))\right] \tag{3.45a}$$

$$L(\phi, \mathbb{B}) = -\mathbb{E}_{\mathbb{B}} Q_\theta(s', \pi_\phi(s')) \tag{3.45b}$$

Minimizing Equation (3.45a) gives more accurate estimates of expected return in taking a particular action in a particular state. On the other hand, minimizing Equation (3.45b) corresponds to choosing the action that maximizes return (or Q-value) from a given state. Improving the policy function directly by performing gradient descent on the function belongs to a class of RL methods called *policy-gradient* methods. This is different from *value-based* methods where the gradient descent is performed on the Q-function. The DDPG actor-critic method combines both value-based and policy-gradient methods. Interested readers can find further details in [OpenAI, 2020b, Lillicrap et al., 2015, Silver et al., 2014].

3.3 Multi-Objective Reinforcement Learning (MORL)

In a general multi-objective problem, an optimizing variable is mapped by *objective functions* into a multi-dimensional objective space. Pareto-optimal points correspond to those mappings where the value of one objective cannot be increased without a decrease in the value of at least one another objective. The overall utility of the

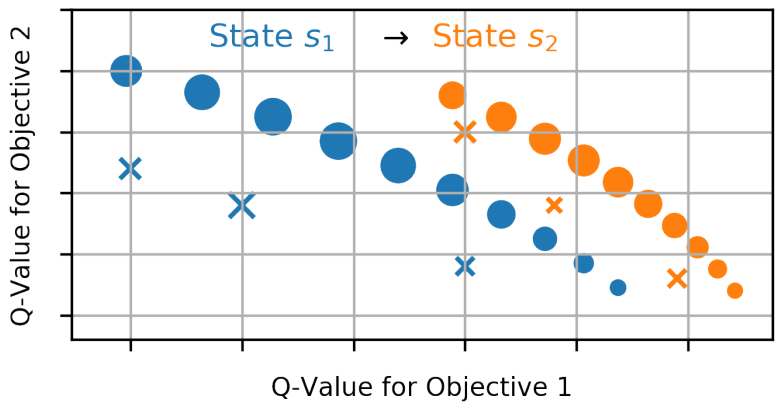

Figure 3-4: The Pareto-front changes when the agent transitions form state s_1 to s_2. Crosses indicate Pareto-dominated actions. The marker size represents the utility of the action.

variable is a linear combination of its mapped objective values according to the user-defined weights (non-linear utility functions are outside the scope of this work).

Generally for a posteriori optimization, Monte Carlo (MC) rollouts [Wang and Sebag, 2012] or Evolutionary Algorithms (EA) [Deb, 2001, Deb et al., 2002, Sindhya, 2011] are used to map a sample population of the optimizing variable to points in the objective space. By doing so, we hope to either find the value of the variable that has the highest utility or approximate the Pareto-frontier with some function or visualization technique. This is possible when the optimizing variables map *statically* to the objective function space. These elitist approaches rapidly grow in complexity as the number of objectives increase.

These methods have been extensively used for optimizing *networks* of sensor nodes w.r.t. coverage, latency, connectivity etc. [Fei et al., 2016]. However, the problem assume a static objective function. For e.g., in [Jia et al., 2009], the authors use GA to minimize energy consumption and node count while maximizing coverage. They assume that the nodes are statically deployed and that the QoS remains constant. Each optimization process takes hundreds of generations to converge. This approach works for this particular problem because the working environment (optimization space) does not vary drastically from one generation to another. However this method is not suitable for EHWSNs where energy variability is generally random and fast

varying. With this method, it would not be feasible to evaluate multiple rollouts and generations from which to select the optimal duty cycle at *each timestep*.

In our MORL framework, the optimizing variable is the agent's action a w.r.t. its policy π. Since we are concerned with the long-term effects of the policy, the objective function is the Q-function $Q^{\pi}(s, a)$, which represents the expected cumulative reward of that action (and not the instantaneous reward). This means that the objective function is not static because $Q^{\pi}(s, a)$ is dependent on the state s, which changes at every timestep. This is shown in Figure 3-4 where the same set of sampled actions map to different points in the objective space when the state changes from s_1 (blue) to s_2 (orange). Due to this non-static mapping, MC/EA methods are too costly to be feasible.

In general, MORL solutions need to learn the Q-functions and a policy that can infer the optimal action from Pareto-frontier. A general Multi-Objective Markov Decision Process (MOMDP) for MORL with m objectives is defined by $(\mathcal{S}, \mathcal{A}, \mathbb{P}, \vec{R}, \gamma, \Omega)$. The reward function is now a vector function, $\vec{R}(s, a) = [R_1(s, a), R_2(s, a), ..., R_m(s, a)]$ and the discount factors for each objective are given by $\gamma = [\gamma_1, \gamma_2, ..., \gamma_m]$. Ω determines the space of preferences among the m objectives. The MORL policy μ is now associated with an m-dimensional vector Q-value function $\vec{Q}^{\mu} = [Q_1, Q_2, ..., Q_m]$ whose components correspond to the different objective functions. μ extracts the Pareto-optimal action w.r.t. the user-preference $\omega \in \Omega$. In Chapter 6, we develop a suitable MOMDP for EHWSNs and present solutions to learn the vector Q-functions and extract the optimal actions.

Chapter 4

General Framework for Adaptive Policies

In this chapter, we present an adaptive power manager for solar energy harvesting sensor nodes. We use a simplified model consisting of a solar panel, an ideal battery and a general sensor node with variable duty cycle. Our power manager uses Reinforcement Learning (RL), specifically SARSA(λ) learning, to train itself from historical data. Once trained, we show that our power manager is capable of adapting to changes in weather, climate, device parameters and battery degradation while ensuring near-optimal performance without depleting or overcharging its battery. Our approach uses a simple but novel general reward function and leverages the use of weather forecast data to enhance performance. We show that our method achieves near perfect energy neutral operation (ENO) with less than 6% root mean square deviation from ENO as compared to more than 23% deviation that occur when using other approaches.

4.1 Introduction

We propose a power management policy based on tabular SARSA(λ) reinforcement learning (RL) [Sutton et al., 1992] for a solar EHWSN. In this method of RL, some fraction of the reward is back-propagated to all the actions and states that contributed

to its achievement. This back propagation of feedback results in faster learning [Sutton et al., 1992] as compared to other methods that propagate their rewards by only one step. In our formulation of the problem, the node is able to dynamically adapt its power consumption depending on the energy harvesting opportunities by varying its duty cycle to ensure ENO. The power manager makes its decisions based on information about its *distance from energy neutrality, battery level, amount of energy being harvested* and *the weather forecast for the day.* The learning theory and its implementation methodology are explained in more detail in Section 4.3 and Section 4.4. The distance from energy neutrality is referred here as Energy Neutral Performance (ENP). One of our main contributions is the inclusion of ENP in the state definitions. This dramatically reduce the learning time and enables the agent to be highly adaptive to environmental changes. Our results, described in Section 4.5 show that the agent is able to adapt to seasonal variations, changes in climate due to change in location, battery degradation and changes in device parameters. As to our current knowledge, the general adaptivity of a RL based power management strategy for EHWSN has not been investigated before.

The reward function is extremely critical in a RL framework as it is responsible for educating the agent on what kind of behaviors are favorable. [Blasco et al., 2013] and [Ortiz et al., 2016] use metrics related to data transmission as rewards. This makes sense when the power consumption of data transmission is significantly greater than that of sensing operations. This assumption limits the scope of application and excludes situations where other critical tasks consume significant power. For example, when a mobile sensor node has to allocate power consumption between locomotion, processing, transmission and sensing, the amount of data transmitted may not be a good basis to determine the optimality of the actions.

The reward function mentioned in [Hsu et al., 2014] is energy-centric has a more general scope of application. Here, the authors propose a reward function based on the instantaneous battery level and ENP. However, the authors use significant reward shaping and the resulting reward function is complex and not intuitive. Furthermore, this reward function fails to learn optimal policies when we change the system pa-

rameters such as battery size, harvesting rate etc.

The reward schemes used in previous RL methods were customized to work with a certain type of task optimization, or tuned for a specific device configuration to operate in a specific environment. These schemes are not 'general' in the sense that the same reward function cannot be used in EHWSNs without requiring significant fine-tuning depending on the application context. This defeats the idea of a unified learning framework for EHWSNs, which is one of the core motivations to use learning-based methods for ENO.

In this work, we improve upon the existing methods and introduce a simplified *general* reward function for ENO that is natural, intuitive and performs satisfactorily. Our reward function is independent of the device parameters such as battery size, harvesting rate and node power rating. With our proposed reward scheme, the agent is able to learn energy neutral policies independent of the energy-harvesting profile and thus independent of the climate and weather. Furthermore, our reward scheme allows the node to adapt to changes in the device parameters (e.g., node/battery degradation) and the environmental conditions. This is an important point because it also means that this reward function is suitable for learning energy neutral policies in different types of EHWSNs with different batter capacities, and power ratings. We use the ENP at the *end of an episode(day)* as the sole basis of our reward function. This is a novel contribution of our work. An added benefit to this is that this reduces the variance in battery levels and this in turn contributes to increase in battery life [Benini et al., 2001]. The novelty of this reward definition is that it is indifferent to the node's tasks that consume power. As a result, our proposed power management policy has a wider scope of application. Here, we assume all the power consumed by the node is for sensing purposes. This is only for the sake of example and we can extend this to any kind of node operation.

Our final contribution is to take into account the information about the weather forecast to enhance performance. [Sharma et al., 2010a] use forecast data to model a weather predictor. It is obvious that the availability of a perfect weather oracle would greatly enhance the performance of a power manager. [Kansal et al., 2007] and [Blasco

et al., 2013] mention some power management strategies when such non-causal data is available. We use the formulation in [Kansal et al., 2007] to obtain the theoretical upper limit in performance and use it for comparison. It is impossible to expect such non-causal information in practice. However, it is possible to obtain some general indication of future weather for the day. The information about the type of weather - for e.g., sunny, fair, overcast etc. can be easily acquired from weather websites and apps. We make use of this information to increase the performance of our agent. For the system presented here, we use the forecast data to give an estimate of expected solar energy. However, the same argument can be made for other EHWSN such as those driven by wind power.

In summary this contribution of our work emphasizes the following points:

- **Adaptivity:** We propose a SARSA(λ) RL based power management approach to ensure ENO. This approach enables the power manager to adapt to changes in weather, climate, battery degradation and device parameters while achieving faster convergence. This is attributed to our novel idea of using ENP as a state definition parameter.

- **General Scope of Application:** A direct consequence of our novel reward function is that the power manager policy can learn to accommodate any kind of realistic system. As a result, the scope of application is more general. Using our method, the policy can adapt accordingly and maintain near-optimal performance even if the system parameters are to change over time.

- **Enhanced performance:** Our power management strategy improves performance by as much as four times by leveraging information about the general weather forecast. The use of such data allows the agent to anticipate the amount of energy that can be harvested and adopt a suitable power management strategy.

The adaptive nature of our learning-based approach coupled with the abstract general system model and MDP formulation makes our proposed solution applicable

to a variety of different usage scenarios. Although we evaluate using only a single system, the results show that our solution can adapt to changes in weather, location. Furthermore, since our core motivation is concerned with long-term operation of the sensor nodes, we also verify that our solution can adapt to node-degradation that may result due to extended operation periods. Specifically we show that our solution can remain energy neutral while adapting to reduced battery capacity, decrease in solar panel capacity and power efficiency of the node. This indirectly indicates that our solution is actually suitable to a variety of EHWSN application scenarios irrespective of the actual hardware specifications. For instance, the same learning framework is able to derive optimal ENO policies for systems that may have different energy-harvesting profiles, batteries and power ratings. This eliminates the need for application-specific hand-tuned algorithms and enables a general learning-based solution for virtually any type of EHWSN capable of duty-cycling.

4.2 System Model

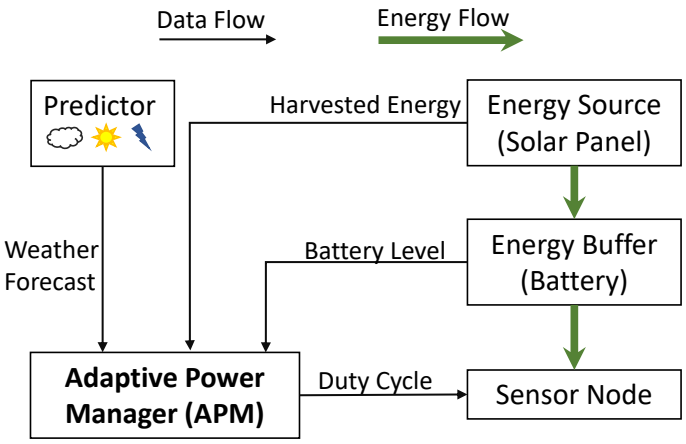

Figure 4-1: The ENO-RL system for a solar-EHWSN. The Adaptive Power Manager (APM) uses RL to regulate the node's energy consumption via duty cycling.

We consider the ENO-RL System (Figure 4-1) that consists of a generic sensor node powered by a battery that is charged by a solar panel. The node's power consumption is determined by its operating duty cycle that is regulated by the Adaptive

Power Manager (APM). We further assume that a higher duty cycle implies a higher power consumption and higher performance from the sensor node. The APM uses RL to learn power management policies for duty cycling to achieve ENO. The APM also takes into account a rough weather prediction when choosing the duty cycles.

4.2.1 Maintaining ENO

We assume the ENO-RL system is equipped with an ideal battery of capacity b_{max}. The APM is capable of varying its power consumption via N_D discrete duty cycles $d_t \in [d_{min}, d_{max}]$, and can harvest a maximum of h_{max} energy per timestep t. We further assume that all of the harvested energy is first stored in the battery before being consumed, i.e., harvest-store-use system. At time t, the reserve battery level is $b_t \in [0, b_{max}]$, the harvested energy is $h_t \in [0, h_{max}]$ and the energy consumed by the node is $z_t = d_t \times z_{avg}$ where $0 < d_t \leq 1$ and z_{avg} is the average energy consumption of the node. The battery at the start of $t + 1$ timestep is given by:

$$b_{t+1} = b_t + h_t - z_t \tag{4.1}$$

The least energy that can be consumed when the node is operational is $z_{min} = d_{min} \times z_{avg}$ corresponding to the lowest duty cycle. The node is operational at time t if $b_t \geq z_{min}$. *Downtimes* are instances when $b_t < z_{min}$ and the node is out of operation. We assume that higher duty cycles imply higher node utility. Energy is irrecoverably lost and *overflow* occurs when $b_t \geq b_{max}$.

We now define the ENO condition for the EHWSN. ENO requires that the amount of energy harvested equals the amount of energy consumed by the node in some time interval. This ensures that the node always has enough energy to operate (no downtimes) and harvested energy is not wasted due to overflow. The objective of the APM is therefore to optimize the duty cycles of the node such that downtimes and overflows are minimized and the utility (energy consumption) of the node is maximized. We assume no losses due to battery inefficiencies. This can be assumed without loss of generality because any such losses can be lumped together as increase

in node power consumption. Similarly the power consumed by the APM can also be lumped with the node power consumption. This simplifies the objective of ensuring ENO to:

$$b_t + h_t > z_{min} \tag{4.2}$$

$$b_t + h_t - z_t < b_{max} \tag{4.3}$$

From the above equations we can conclude that as long as $z_{min} < b_t < b_{max}$ for all t, the node is consuming all of the energy that it has harvested and is therefore energy neutral (assuming no downtimes or overflows). The ENP gives a measure of the deficit or surplus of energy required to maintain ENO. If we assume that there are no downtimes or overflows, then the ENP at time t is given by $p_t = b_t - b_{init}$ where b_{init} is the battery level at the beginning of an episode E.

It must be noted that during some episodes, the total harvested energy may not be sufficient to power the node even at its lowest duty cycle. Alternatively, sometimes the energy harvested may be so plentiful that even with the highest duty cycle, some surplus energy would remain unused. The value of b_{init} plays an important role in such cases to ensure ENO. Although it is difficult to express b_{init} formally to guarantee ENO, in practice, the range of suitable b_{init} values can be roughly estimated using guidelines in [Kansal et al., 2007]. Here, we assume that as long as the maximum deviation of b_{init} from b_{opt} is less that $b_{margin} \geq 0$, i.e., $|b_{init} - b_{opt}| \leq b_{margin}$, there exists a power management policy to achieve ENO. b_{opt} and b_{margin} are hyperparameters that can be estimated using the formulations in Section 3.1. Table 4.1 lists some of the key terms used in this work.

Table 4.1: Key Terms Used

Term	Description	Term	Description
b_t	Battery Level	s_t	state
d_t	Duty Cycle	a_t	action
h_t	Harvested Energy	r_t	reward
z_t	Node Energy Utilization	π	policy
p_t	Energy Neutral Performance	m_t	experience
f_E	Weather Forecast	ϵ	exploration rate

4.3 Proposed SARSA Framework

In this section, we explain how we set up the RL framework for our problem. We also present the algorithm for SARSA(λ) learning.

4.3.1 Defining the Markov Decision Process

The MDP defines the state space, the action space and the reward function. The environment, in this context, consists of a stochastic energy source and the battery. It reacts to the agent by specifying a reward based on the duty cycle chosen and a new state depending upon the amount of energy harvested, the reserve battery level and the weather forecast information.

We use an episodic discounted MDP in this framework and train the agent in episodes. The agent takes a sequence of actions and traverses through a series of states until the end of the episode. At the end of the episode, the agent is awarded some reward which it uses to evaluate its actions and upgrade its Q-table. The state definitions, action space and the reward scheme are described as follows.

State Space

Given the statistics of the environment, the optimum battery level b_{opt}, can be determined by using the formulation in Section 3.1. What this means is that if the agent starts out with b_{opt} amount of battery at the start of every day, it is able to accommodate for the days with the least energy harvested as well as days with the

maximum energy harvested without depleting the battery completely or allowing it to be overcharged. We use this battery level to calculate the distance from energy neutrality. Ideally, we would like the agent to begin everyday with b_{opt} amount of battery and end with exactly b_{opt} remaining. If the battery level at time t is b_t, we calculate the distance from energy neutrality or ENP, p_t, using b_{opt} as reference according to Equation 4.4.

$$p_t = b_t - b_{opt} \tag{4.4}$$

The different states in which the agent can exist is given by combination of $S_{batt}(t)$, $S_{dist}(t)$, $S_{eharvest}(t)$, and $S_{day}(t)$ i.e.,

$$(S_{batt}(t), S_{dist}(t), S_{eharvest}(t), S_{day}(t)) \in S$$

p_t is used to determine $S_{dist}(t)$, the state of distance from energy neutrality. Using this information for state definition makes our approach largely independent of the actual battery capacity and thus more generally applicable. However, the agent also takes into account whether the battery is in danger of being overcharged or completely depleted. This is reflected in $S_{batt}(t)$, the state of the battery reserve level. State $S_{day}(t)$ gives an indication of what kind of weather the agent may expect. With this state information, the agent is able to use different strategies to achieve ENO depending on how much energy it can expect to harvest. Finally, the state of the amount of energy being harvested during time t is given by $S_{eharvest}(t)$.

Action Space

The action space, A, $|A| = N_D$, $A \in (d_{min}, d_{max})$, is defined as the set of discrete duty cycles that can be chosen by the APM. d_{min} and d_{max} are the minimum and maximum duty cycle of the sensor node. The agent chooses one duty cycle in each timestep.

Reward Function

The reward indicates what kind of behavior best serves our objective. In our RL formulation, the reward awarded at the end of an episode depends simply on ENP at the end of the episode. We call this quantity Terminal Energy Neutral Performance (TENP). The "goodness" of any action in the middle of an episode cannot be judged without taking the effect of other actions in the episode. Hence we wait until the end of the episode to judge the "goodness" of the sequence of actions chosen. This means the agent is rewarded only once during an episode (as opposed to being rewarded at every timestep). This sparse rewarding scheme ensures that the agent only cares about the TENP (and not the ENP at every timestep). This is reasonable because ENO can be achieved by a number of different methods i.e., it is not always necessarily the case that there is only one unique optimal duty cycling policy. This is because we have not included battery inefficiencies in our formulation. As a result, the system is indifferent to using energy directly from the solar panel or from the battery. However, the sparsity of rewards also makes it very difficult for the agent to converge to optimal policies.

Ideally we would like the agent to learn a policy to ensure zero deviation from its optimal battery level at the end of each episode. Lower deviation are awarded larger rewards and viceversa.

4.3.2 SARSA(λ)

The reward at the end of an episode is used to update the Q-values of the state-action pairs according to Algorithm 2 [Sutton et al., 1992]. The use of eligibility traces allows us to propagate the reward backwards to all the state-action pairs that contributed to it. Before the training, the Q-table is optimistically initialized to a high value *qinit*. At the start of each episode, the eligibility traces are reset and an action is selected randomly. The agent then determines its state and uses an ϵ-greedy policy to interact with its environment until the end of the episode. The reward it receives at the end of each episode is used to update the Q-table towards a better estimate.

4.4 Evaluation Setup

Here we describe the specifications of our system model and the parameters that are used during training and implementation.

4.4.1 Simulation Environment

We base our system specification on a scaled up version up of a TMote Sky node [TMote, 2019]. This mote is powered by a 3.6V, 2200 mAh NiMh battery and a 6 W solar panel (220mm $\times$ 175mm). The power consumption of Tmote varies from approximately 100 mW to 20 mW depending on the type of task. To ensure stability of the learning process, we scale up these values roughly by a factor of five. We chose this particular configuration simply for the sake of example.

Energy Source:

To simulate the solar energy harvested, we acquire global solar radiation data from the Japanese Meteorological Agency (JMA) website `http://www.jma.go.jp`. JMA provides hourly data on the global solar radiation several locations in Japan. As a result, one timestep corresponds to one hour. We use the solar radiation data to calculate the electrical energy harvested by the solar panel.

Sensor Node:

The scaled up values of the TMote correspond to a sensor node that consumes energy varying from 100 mWh to 500 mWh during each timestep according to the specified duty cycle. We assume the power consumption remains constant within each timestep. The possible duty cycles are 20%, 40%, 60%, 80% and 100%. We do not consider sensing latency inherent in sensor nodes for sake of simplicity. The power manager is indifferent to how the node allocates its power consumption.

Battery:

Using the guidelines in [Kansal et al., 2007], we calculate the battery size, b_{max}, and the optimal initial battery level, b_{opt}. Using statistics for the year 2010, we arrive at $b_{max} = 40000mWh$ and $b_{opt} = 60\%$ of b_{max}. We assume an ideal battery for the sake of simplicity. Any inefficiencies of the node, solar panel and the battery can be lumped together as an increase in node power consumption.

4.4.2 SARSA(λ) Parameters

Action Space

The action space, A, defines the actions that can be chosen by the agent. In our case, the action space is $A = \{20\%, 40\%, 60\%, 80\%, 100\%\}$.

State Definitions

The state of the system at time t is given by

$$S_k = (S_{batt}(t), S_{dist}(t), S_{eharvest}(t), S_{day}(t)).$$

The state of the agent is determined from the actual values observed by the agent. Since these values are continuous in nature, it is necessary to discretize and define which values correspond to which states. We use the following to assign states from actual observed values.

$S_{batt}(t) \in \{S_{b1}, S_{b2}, S_{b3}\}$ gives information about whether the agent is in danger of depleting its battery or overcharging it. This is determined by the value of b_t according to Table 4.2.

Table 4.2: $S_{batt}(t)$ Assignment

$S_{batt}(t)$	Range
S_{b1}	$b_t < 20\%$ of b_{max}
S_{b2}	20% of $b_{max} \leq b_t < 80\%$ of b_{max}
S_{b3}	80% of $b_{max} \leq b_t < 100\%$ of b_{max}

$S_{dist}(t) \in \{S_{d1}, S_{d2}, ..S_{d40}\}$ gives information on how far the agent's operation is from ENO. This state is determined by the value of p_t according to the following.

- $S_{dist}(t) \in \{S_{d22}, ..S_{d40}\}$ correspond to states in which $p_t < 0$ i.e., the battery is at a level lower than b_{opt}.
- $S_{dist}(t) \in \{S_{d21}\}$ corresponds to the state of perfect energy neutrality i.e., $p_t = b_t - b_{opt} = 0$;
- $S_{dist}(t) \in \{S_{d1}, S_{d2}, ..S_{d20}\}$correspond to states in which $p_t > 0$ i.e., the battery is at a level higher than b_{opt}.

Neighboring states are spaced at a distance of 1000 mWh from each other.

$S_{eharvest}(t) \in \{S_{e1}, S_{e2}...S_{e7}\}$ gives information about how much energy is being harvested at time t. Its value is determined by the amount of solar energy harvested h_t, according to Table 4.3.

Table 4.3: $S_{eharvest}(t)$ Assignment

$S_{eharvest}(t)$	Range
S_{e1}	$h_t = 0mWh$
S_{e2}	$0mWh < h_t \leq 100mWh$
S_{e3}	$100mWh < h_t \leq 500mWh$
S_{e4}	$500mWh < h_t \leq 1000mWh$
S_{e5}	$1000mWh < h_t \leq 1500mWh$
S_{e6}	$1500mWh < h_t \leq 2000mWh$
S_{e7}	$h_t > 2000mWh$

$S_{day}(t) \in \{S_{f1}, S_{f2}...S_{f6}\}$ gives the agent general information about what kind of weather it can expect during the day as shown in Table 4.4. In real world application, this information can be obtained from weather apps or websites. For our case, we differentiate each day into one of six different types according to the value of e_{day}, the total amount of energy harvested in that particular day (or episode). Since we are training on historical information, e_{day} can be easily calculated prior to the training by using Equation 4.5. $S_{day}(t)$ remains constant for each day (episode).

$$e_{day} = \sum_{k=1}^{24} h_t \tag{4.5}$$

Table 4.4: $S_{day}(t)$ Assignment

$S_{day}(t)$	Weather	Range
S_{f1}	Very little sun	$e_{day} < 2500mWh$
S_{f2}	Overcast	$2500mWh \leq e_{day} < 5000mWh$
S_{f3}	Partly Cloudy	$5000mWh \leq e_{day} < 8000mWh$
S_{f4}	Fair	$8000mWh \leq e_{day} < 10000mWh$
S_{f5}	Sunny	$10000mWh \leq e_{day} < 12000mWh$
S_{f6}	Very Sunny	$e_{day} \geq 12000mWh$

Reward Function

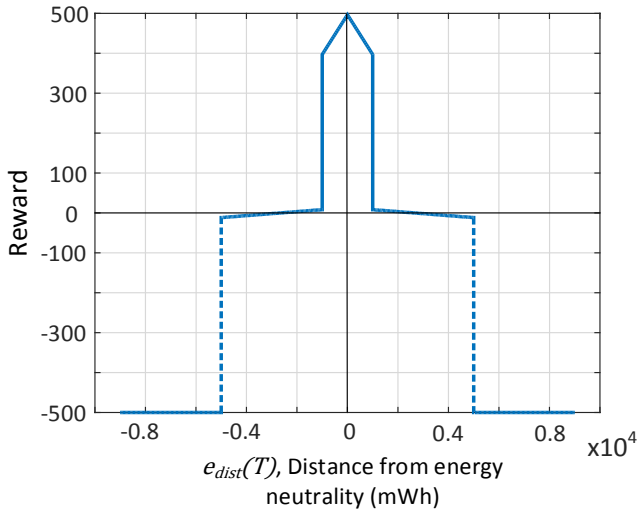

Figure 4-2: Reward Function

The reward function in Figure 4-2 shows the relationship between the reward and TENP. Since TENP cannot always be exactly zero, we consider a margin of $b_{margin} = \pm 1000$ mWh deviation as acceptable. The symmetry of the reward scheme is due to the fact that both positive and negative deviations from ENO have the same effect on the reward. During training, the rewards oscillate around some final value which introduces some noise in the reward. The sharp profile of the reward function allows the learning algorithm to arrive at true Q-values in spite of this noise. The gradual slope between ± 1000 mWh and± 5000 mWh deviation enables the agent to incrementally improve its performance without causing unstable oscillations during learning. This reward function is a novel contribution of our work. It is simple,

intuitive and best reflects the ENO objective. Moreover, this reward function is independent of system model specifications and thus has a general scope of application.

4.4.3 Training Parameters

We use historical data from JMA [Japan Meteorological Agency, 2019] for training the agent. The solar radiation data is provided on an hourly basis. So each timestep is an hour long with one episode consisting of 24 timesteps (hours). The agent iterates $N_l = 10^6$ times for *each day* using the SARSA(λ) learning algorithm.

The agent is trained in three phases. In the first phase of training, the initial battery level is initialized to some middle value. In the second and third phase of training, this is initialized to high and low values respectively. We use this training scheme so that the agent is exposed the different regions of the state-space that would not have easily accessible using ϵ-greedy exploration methods. By maximizing the state-space coverage, the agent is able to ensure ENO even when battery levels are not initialized at optimal levels. For the three phases of training, we fix the initial battery level b_init at 60%, 80% and 20% of b_{max}.

We learn using Algorithm 2 with $\alpha = 0.001$, $\gamma = 0.8$ and $\lambda = 0.3$. As with most RL problems, these values were determined empirically rather than through mathematical methods. We evaluated the system with different combinations of the values of the hyper parameters and chose the combination that performed the best. The system is somewhat sensitive to the values of α and ϵ. Using a high values for α (learning rate) and ϵ causes large oscillations in Q-values during training. Hence, we chose smaller values but compensated with a larger number of iterations during learning.

We train our agent with weather data of Tokyo for the year 2010. We then observe its performance for the year 2011 in Tokyo and Wakkanai. Wakkanai lies in far north of Japan and experiences a drastically different weather than that of Tokyo. We use this change in location and climate to observe the node's adaptive behavior.

4.4.4 Evaluation Metrics

We refer to our power management policy as *SARSA Policy*. We compare our policy to a power management strategy mentioned in [Kansal et al., 2007] referred here as *Offline Policy*. Offline Policy uses linear programming optimization methods with non-causal data on energy harvesting opportunities to determine the optimal duty cycles. The results of this method are presented here as an estimate of the upper limit of performance. This is not a practical method of power management because non-causal data on energy harvesting opportunities is not available in real life. The Offline Policy solutions have real continuous values and therefore to ensure fair comparison, the values of the duty cycles are rounded off to the nearest possible duty cycle of the system. As a result of this rounding off process, the Offline Policy does not always achieve perfect ENO. The Offline Policy uses an optimization window of one day (24 hours) to calculate the duty cycles. This also ensures fair comparison with our method because our SARSA(λ) agent is also trained in one-day episodes.

We express the battery levels, duty cycles and energy harvested in percentage of their maximum values. ENP is expressed as a percentage of the maximum battery value b_{max}. We use Root Mean Square (RMS) values of ENP to compare performance between different policies. Lower RMS values indicate lower swings in the battery levels which correspond to more robust energy-neutral policies.

4.5 Results

4.5.1 Learning Convergence

Figure 4-3 shows the convergence of SARSA(λ) to its final optimal policy as the agent trains over a number of iterations. The graph shows the end of the day deviations from the initial optimal battery level, b_{opt} for each iteration during training on days 58 and 69 of the year 2010 (Tokyo).

We see that the agent is learning vigorously during the first 30000 to 40000 iterations. After 50000 iterations, it has figured out the optimal policy as is evidenced by

the low deviation from energy neutrality in the figure. For the next 50000 iterations, it follows this policy greedily taking an occasional random action to see if it might lead to a better policy.

The fluctuations in the deviation from energy neutrality is the consequence of the limited number of possible discrete power consumption modes in the node. Since the harvested energy can take continuous values, it is highly unlikely that perfect energy neutral operation will ever be achieved. There will always be a small amount of error. We allow an error margin of ±1000 mWh (±2.5% of b_{max}) for the best case scenario. We tolerate up to ±5000 mWh (±12.5% of b_{max}) of deviation. This is also reflected in the reward function shown in Figure 4-2. Also, a small change in one of the node's actions may cause the deviation to jump between positive and negative values. This may explain the oscillation in deviation values as the agent converges to an optimal policy.

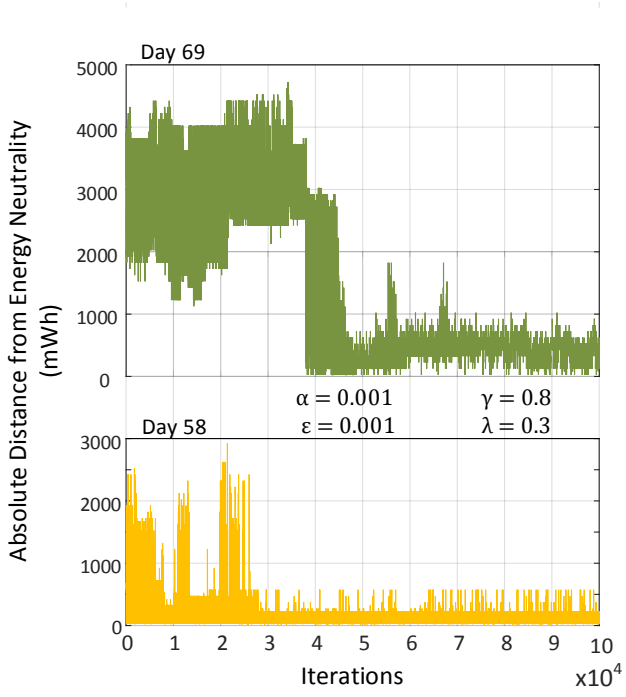

Figure 4-3: Policy Convergence

4.5.2 Energy Neutral Operation

Figure 4-4 compares the SARSA Policy with Offline Policy and Constant Duty Cycle (CDC) Policy for January 29, 2011 Tokyo. The CDC Policy maintains a constant node power consumption rate. We can determine the constant duty cycle to ensure ENO by dividing the total energy harvested during the day by the total number of hours. Of course, this constant duty cycle is not a realistic policy because it requires non-causal information. We illustrate it for comparison purposes only.

It is worth noting that although the duty cycles of the Offline Policy and SARSA Policy differ from each other, they both have very similar performance. Both start the day off with 60% battery end with near about the same battery level. The green dotted-dashed line indicates the initial level of the battery ($B_{init} = b_{opt}$). The ENP at the end of the day for SARSA Policy and Offline Policy was 491.875 mWh and 191.875 mWh respectively. SARSA Policy violated the energy neutrality by only 300 mWh (0.75% of b_{max}) more than that of Offline Policy. In fact, SARSA Policy shows slightly lesser RMS deviation (5.03%) as compared to the Offline Policy (5.06%). This shows that SARSA policy comes very close to optimal performance using only a general knowledge of the weather forecast.

As mentioned before, the Offline Policy optimizes its performance when considering one day at a time. In Figure 4-5, we compare SARSA Policy with the Offline Policy with a one-day window (shown in violet) as well as with an Offline Policy with a 30-day window (shown in blue) for 2011, Tokyo. We allow the duty cycles of the Offline Policy with a 30-day window to have continuous values between 10% and 100% and as a result, perfect energy neutrality is achieved using this policy.

We observe the battery profiles for different polices for a 30-day period. We observe that both SARSA Policy and one-day window Offline Policy have similar behavior and try and maintain the battery level at 60% of b_{max} at the end of every day. In contrast, the Offline Policy with a 30-day window deviates significantly from the optimal battery level during the middle of the 30-day period. The SARSA Policy can also be modified to optimize for a longer window. Such longer windows optimization

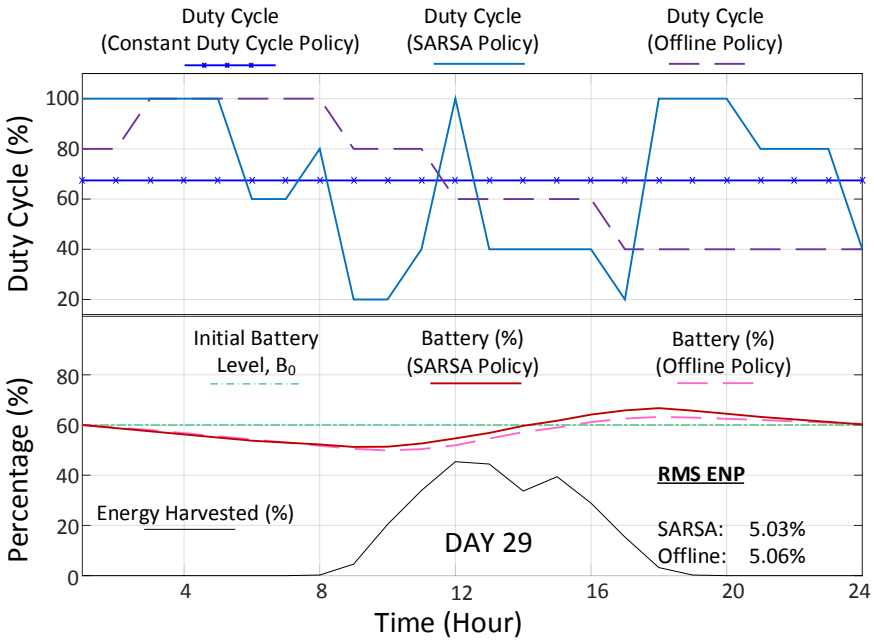

Figure 4-4: SARSA and Offline Policy

techniques show better performance especially when the battery is nearing its limits. However they require more computation and exponentially longer training sessions.

Figure 4-5 also shows the battery profiles for two other policies - the CDC policy (shown in dotted blue) and a naïve battery-centric policy (shown in dashed orange). The CDC policy in this case uses a constant duty cycle that is obtained by averaging the total energy harvested over a 30-day period. This policy is not at all adaptive and therefore may lead to battery overcharge/depletion and ultimately wastage of energy and node failure. We can observe that some amount of energy is wasted due to overcharging during Day 286 as a result of this policy.

The Naïve policy is the simplest adaptive policy. It is battery-centric in that the duty cycle is proportional to the battery reserve level. Higher battery drives the node with higher duty cycles and vice versa. While this policy is simple to implement, it is not very intelligent. For instance, on Day 288, the Naïve policy deviates quite far from the optimal battery level and drops to almost 20% on Day 292. If the days following Day 292 were not sunny enough, the battery could very well have been depleted. Figure 4-6 shows the RMS deviation from energy neutrality at the end

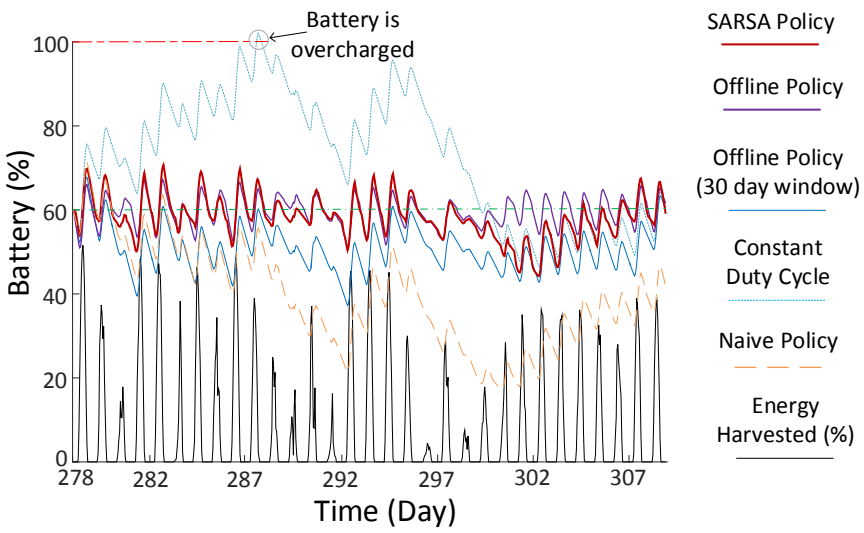

Figure 4-5: Comparison of different policies

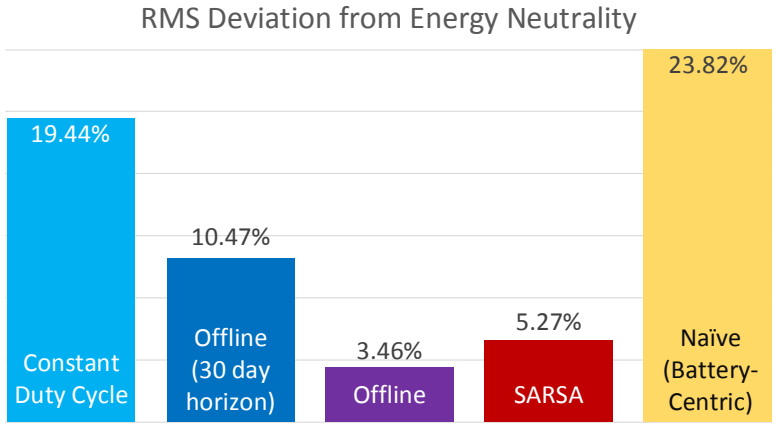

Figure 4-6: Energy neutral performance of different policies

of the 30 day period. We can see that the Naïve Policy suffers from the largest deviation at more than 23%. SARSA Policy and one-day window Offline Policy show very little deviation 3.46% and 5.27%. The constant duty cycle and 30-day horizon Offline policy exhibit perfect energy neutrality. However this comes at the cost of larger battery fluctuations as illustrated in Figure 4-5.

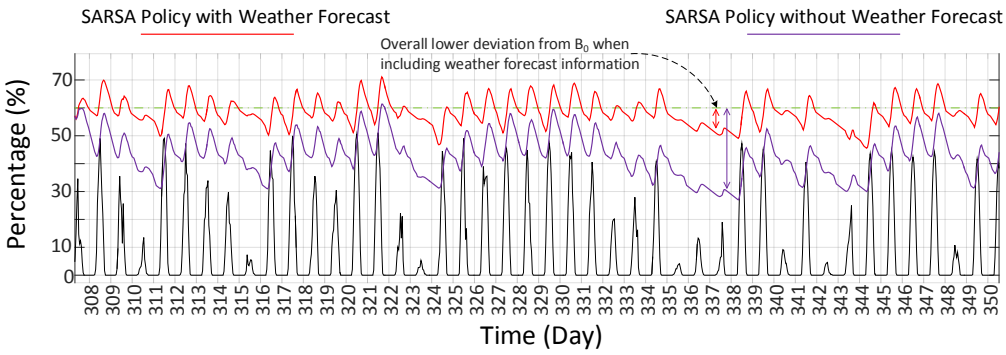

Figure 4-7: Effect of weather forecast information

4.5.3 Effect of including weather forecast

In Figure 4-7 we compare the behavior of SARSA policies with and without forecast information from Day 308 to 324 (Wakkanai 2011). The policy that does not involve weather forecast in its decision making (violet) deviates more from the optimal battery level (shown in dotted-dashed green) than the policy that considers weather forecast information (shown in red). Both policies strive to achieve energy neutrality but the policy with weather forecast is more successful because it leverages on the weather forecast information to make better decisions. On the other hand, the weather agnostic policy uses a general policy for all weather types and this leads to lower performance in comparison.

4.5.4 Adaptation to Seasonal Changes

Figure 4-8 and 4-9 compare the SARSA Policy and Offline Policy for a week in spring and winter of Tokyo 2011. Figure 4-8 shows the plots for the week starting from February 27. The second and third day receive very little sunshine. However, starting from the fourth day, it gets a lot sunnier. We can see how SARSA adapts its strategy when it is anticipating a sunny or non-sunny day. We also observe that for the last three days of the week, both SARSA Policy and Offline Policy max out the duty cycle to 100% to be able to use all of the energy being harvested. In spite of their best efforts, the amount of energy that is harvested exceeds what can be consumed. This surplus is stored in the battery and is reflected in the rising battery level.

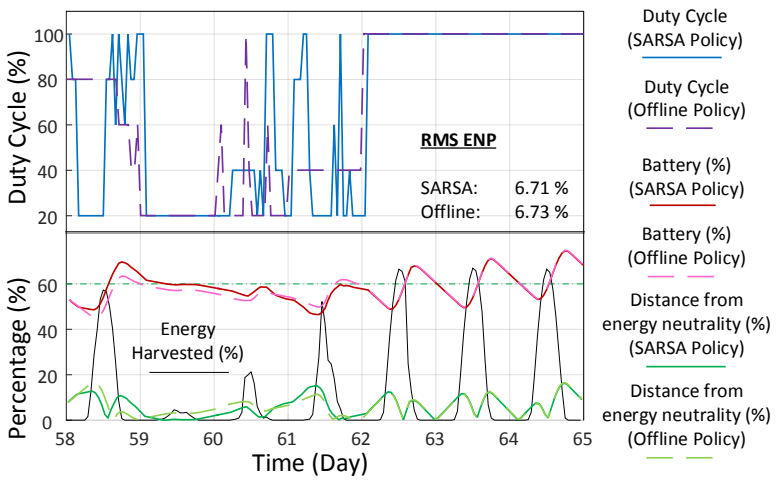

Figure 4-8: Tokyo Spring 2011

Conversely during the week (starting from Nov 29) in winter (Figure 4-9), the amount of energy harvested is not sufficient to sustain even the minimum duty cycle. During the first three days, both policies try to maximize their duty cycle as much as possible. However from the fourth day onwards, both policies fall back to the lowest duty cycle. The last day of the week is quite sunny and so both policies replenish their battery back to the optimum level. We see that SARSA has a similar RMS ENP compared to the Offline Policy during both spring and summer. This shows that the SARSA Policy is able to maintain near-optimal performance while still being able to adapt to seasonal changes.

4.5.5 Adaptation to Climatic Changes

To simulate a change in environment due to change in location, we observe the behavior of the node in Wakkanai, a place with a climate drastically different from which the node was originally trained on. Figure 4-10 shows the performance of both SARSA Policy and Offline Policy for a two-week period starting from February 4. We can see that both policies have similar behavior.

The duty cycles corresponding to SARSA Policy have more variance than that of Offline Policy. This may be due to the fact that Offline policy has perfect prior knowledge of all energy harvesting data and so can optimize its policy better. SARSA

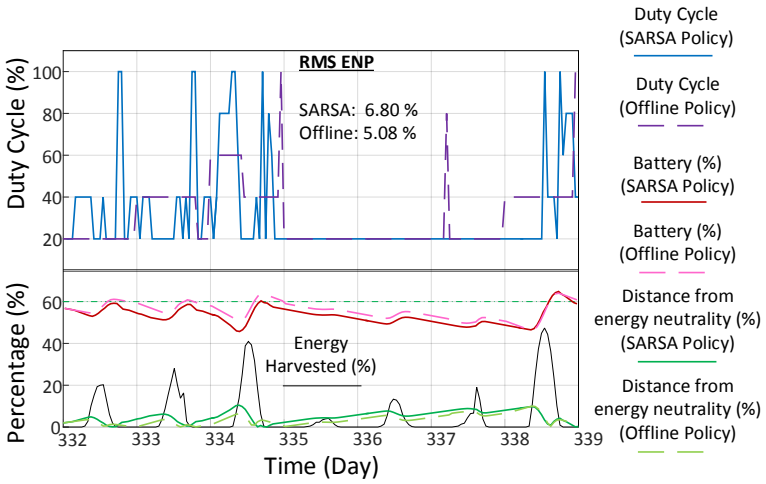

Figure 4-9: Tokyo Winter 2011

Policy on the other hand does not have this information and instead has to decide on an action only after it observes the current harvested energy state. This may also explain why SARSA Policy has higher RMS ENP than Offline Policy.

We believe that SARSA Policy adapts quite well to Wakkanai environment because we have used distance from the energy neutrality as one of the state definition. Since the SARSA Policy is largely independent of the battery level (except when it is near the extreme limits), the strategy it uses during severe Tokyo winters also works for Wakkanai. Figure 4-11 is a color map that shows the deviation from energy neutrality for both policies in the year 2011. The top half shows the results for Tokyo whereas the bottom half is for Wakkanai. Each of the four color maps consists of 12 rows corresponding to each month in the year 2011. Each cell of a month row corresponds to a day in the month and its color indicates the deviation from energy neutrality at the end of that day. Blue cells indicate positive deviation i.e., more energy is harvested than that is consumed. Red indicates that more energy is consumed than that was harvested. Greener the cell, the lesser the deviation. The RMS ENP over the course of the year is also shown alongside the figures.

The battery is initialized to the optimal value at the start of each day to get a fair idea of deviation from energy neutrality for each day. SARSA Policy is able to achieve ENO in most cases. The RMS ENP for Offline Policy gives us an estimate of

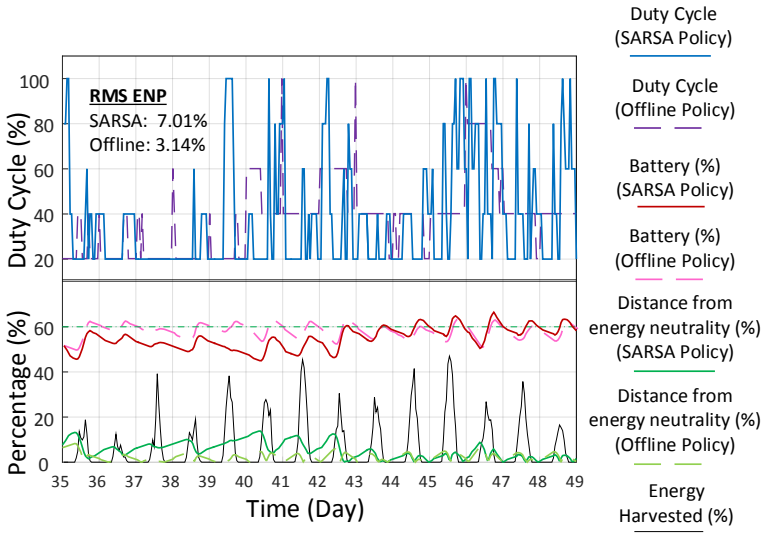

Figure 4-10: Wakkanai Feb 2011

the best possible performance. We observe that SARSA Policy comes very close to achieving optimal performance by adapting to the changes in the environment.

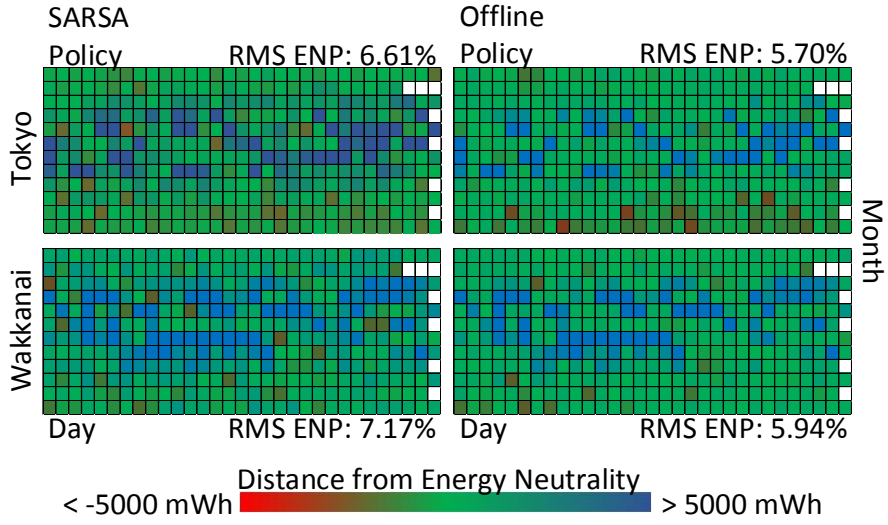

Figure 4-11: Wakkanai vs Tokyo

4.5.6 Adaptation to Battery Degradation

To simulate the degradation of battery of the sensor node, we observe the behavior of SARSA Policy for 2011 Wakkanai weather with only half the original battery

capacity. b_{max} is now reduced to 20000 mWh and $b_{opt} = 60\% of b_{max}$ becomes 12000. The SARSA Policy now tries to achieve ENO around this new b_{opt}. Since the SARSA Policy gives greater weight to the distance of energy neutrality than the actual battery level, the policy is able to achieve energy neutral performance even when the battery capacity is halved. This is shown in Figure 4-12. The RMS ENP for Offline Policy again gives us an estimate of the upper limit in performance. We see that our policy comes very close to achieving this in spite of a drastic change in device parameters. When comparing with the Offline Policy, we see that SARSA Policy show a slightly reduced performance (as evidenced by fewer green cells) but has an overall satisfactory result.

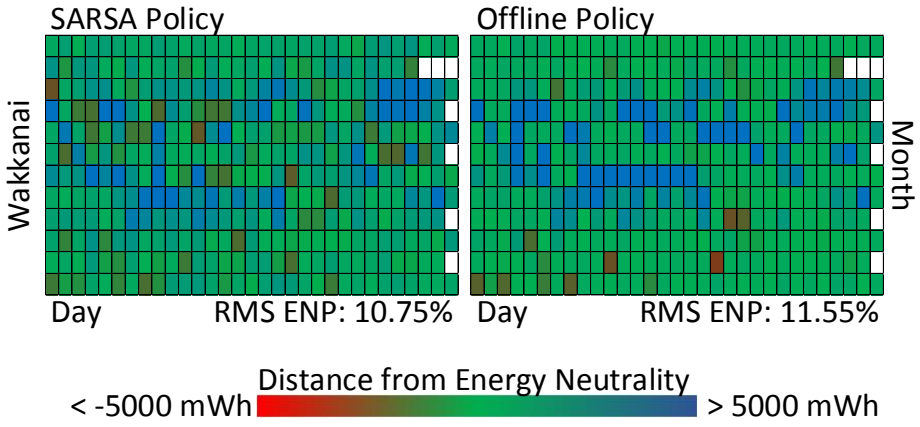

Figure 4-12: Performance with half battery capacity

4.5.7 Adaptation to Changes in Device Parameters

In real life application scenarios, the sensor node may have to accommodate for changes in its device parameters such as a decrease in energy harvesting efficiency and energy efficiency of the sensor node. Our proposed solution is able to adapt to such changes and learn to perform optimally in such scenarios. To simulate such a setting, we observe how SARSA policy learns to adapt to conditions when we

- halve the solar cell output

- increase the node's power consumption by 2.5 times (to simulate a degradation in the node's energy efficiency)

The sensor node (agent) has to be put in a "learning mode" so that it can adapt to these changes in working parameters. In this learning mode, the agent is allowed to train for 1000 iterations with $\alpha = 0.1$ and $\epsilon = 70$ in its new environment. The high learning rate and exploration rate allows it to quickly make changes in its learned Q-table. An additional advantage of this adaptive behavior is that the sensor node can make up for any initial calibration errors by the user. Even if the sensor node is assumed to be deployed in an environment it was not initially designed for, it can quickly adapt and work optimally. Thus, even though the simulation and real world applications may differ, our approach to power management control is able to accommodate these differences and still perform optimally. Any change in this system (for e.g., change in battery capacity or node power consumption) is accommodated for by the power manager due to its adaptive nature. This in fact is the essence of our approach - our power management policy is fluid enough to find a near-optimal policy for any realistic system specification without the need for any intervention by the user.

Figure 4-13 shows the battery profiles for SARSA and Offline Policy with default device settings (without being put into the learning mode) for October 31, 2011. This is similar to the results in Figure 4-4. The only difference is that it is for a different date. We use this as the baseline.

Next, we observe the performance of the SARSA Policy when the solar panel output is reduced by half. This can be due to decrease of solar panel efficiency or a mismatch in design parameters during testing and implementation. The battery profiles for SARSA and Offline Policy is shown in Fig 4-14. We see that SARSA is able to achieve energy neutrality with very little battery deviation at the end of the day. In fact, in this particular case, SARSA policy has a lesser deviation from ENO than Offline Policy. Although the Offline Policy does represent the theoretical upper limit, the rounding off process involved can introduce some errors (as observed in this case).

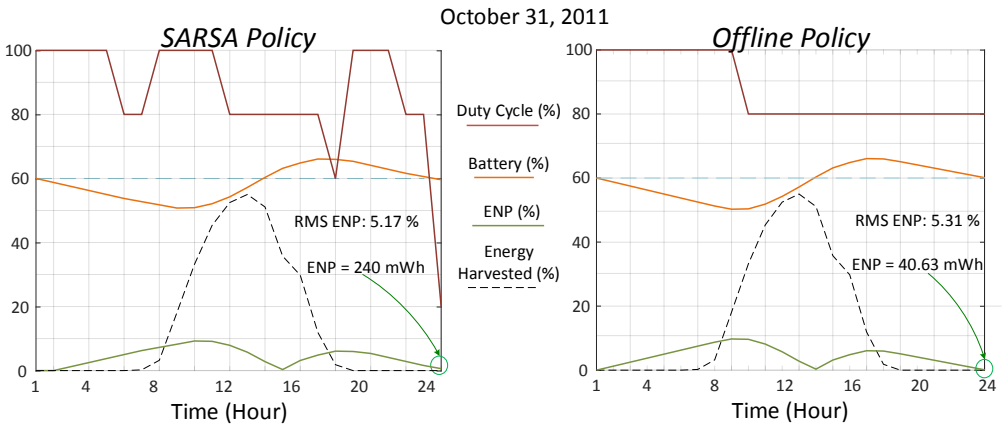

Figure 4-13: Default Device Settings

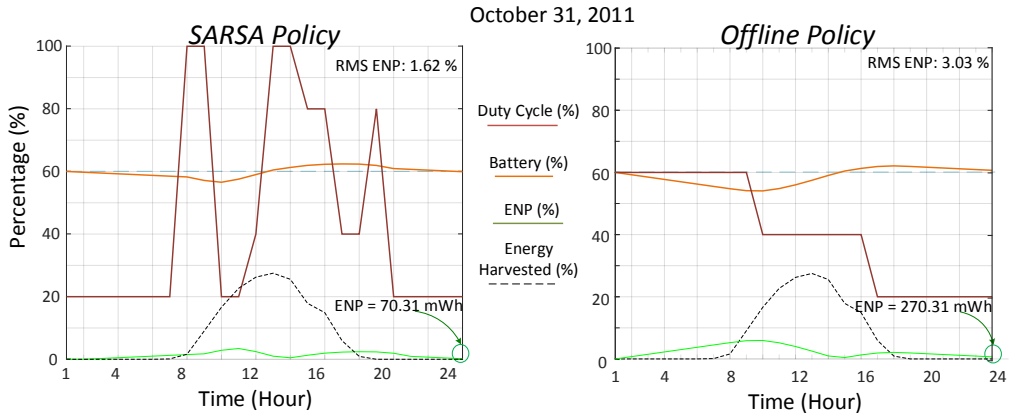

Figure 4-14: Performance with half solar panel capacity

Finally, we consider the case when the sensor node consumes more power than it was initially assumed. This again maybe due to decrease in the node's working efficiency or mismatch in design and implementation. In Figure 4-15, we see that SARSA is able to achieve better performance than Offline Policy in this case also. SARSA achieves an ENP as low as 70.31 mWh at the end of the day.

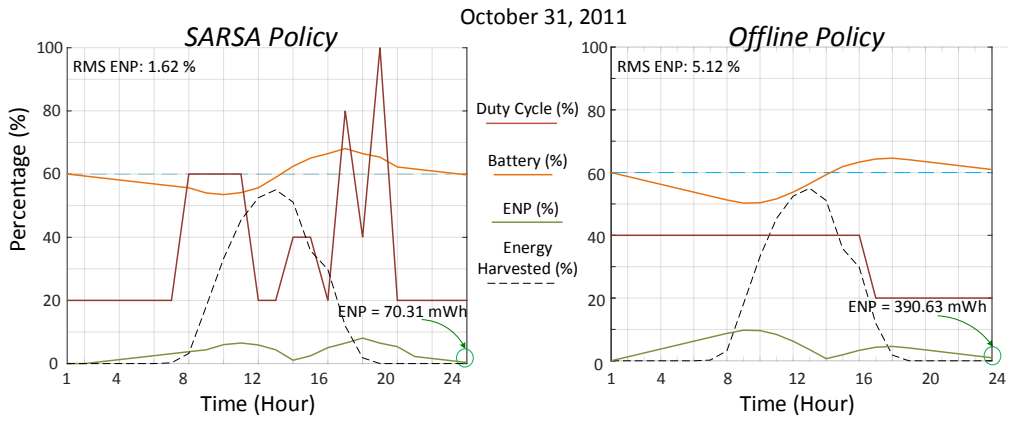

Figure 4-15: Performance with increased node power consumption

4.6 Conclusion

We observe that our SARSA(λ) based RL approach achieves near perfect ENO making autonomous operation a possibility. We also see that our definitions of the state results in a highly adaptive behavior. Our proposed method is able to adapt to changes in weather, location (climate), battery degradation and device parameters which makes the sensor node robust in its operation. In addition, our state definition and general reward formulation scheme allows for general application of our power management method independent of the system it is being implemented in. We also show that inclusion of weather forecast information enhances the performance of the proposed scheme.

Chapter 5

Coordinated Exploration with Distributed Reinforcement Learning (DiRL)

EHWSNs require adaptive energy management policies for uninterrupted perpetual operation in their physical environments. Contemporary online RL solutions take an unrealistically long time exploring the environment to converge on working policies. Our work accelerates learning by partitioning the state-space for simultaneous exploration by multiple agents. We achieve this by using a novel coordinated ϵ-greedy method and implement it via Distributed RL (DiRL) in an EHWSN network. Our simulation results show a *four-fold increase* in state-space penetration and reduction in time to achieve optimal operation by an order of magnitude (50x). Moreover, we also propose methods to reduce instances of disastrous outcomes associated with learning and exploration. This translates to reducing the downtimes of the nodes in simulations corresponding to a real-world scenario by one thirds.

5.1 Introduction

Adaptive power management policies ensure uninterrupted operation of EHWSNs without the need for human intervention even when the working environment is com-

plex and unpredictable. Judicious regulation of duty cycles for ENO has been achieved through Reinforcement Learning (RL) methods [Shresthamali et al., 2017]. This learning-based method is adaptive by nature and dispenses with the need for hand-crafted optimizations which makes it very suitable for a wide range of application scenarios.

During learning, the RL agent has to explore a vast state-action space (for instance, the space of all the combinations of different duty cycles, battery levels and harvested energy) in search of optimal policies. On the other hand, it is also equally important for the agent to maximize its utility by behaving optimally. ϵ-greedy methods are a simple yet practical method to manage the exploration-exploitation trade-off. Using this method, the agent acts greedily according to its learned policy, but with a probability ϵ, it takes a random exploratory action.

Motivation: Ideally we would prefer EHWSNs to acclimatize in their working environment and reach maximum utility as soon as possible. However, acting greedily prematurely leads to weak policies that cannot deal with anomalous and rare states that ultimately reduces the overall utility. Robust policies require longer exploration periods which translates to lost opportunities to maximize utility and higher probability of disastrous outcomes. Here disastrous outcomes refer to battery violations when the node goes out of operation due to insufficient energy (*downtimes*) or when there is an excess of harvested energy that cannot be utilized nor stored in the battery (*overflows*). Downtimes result in reduced coverage and rerouting which directly affects the network quality of service. Overflows decrease the energy efficiency of the node.

Good exploration of the working environment during RL produces better policies. However simply having a higher exploratory rate does not necessarily translate to good exploration. This is because some regions of the state-action space may not be easily accessible through naive undirected exploration. For example, the chances of an agent with very low battery reserves exploring higher states of battery level are extremely slim if the duty cycles are randomly chosen. As a result of insufficient and inefficient exploration, the sub-optimal solutions obtained are not robust. A

possible workaround is to use simulators and historical data with offline training to extensively pretrain the nodes. However this is not always a feasible solution, especially for unknown environments.

Contribution: In this work, we propose to coordinate the many nodes of the sensor network to explore different regions of the vast problem state-space. For this, we base our work on a conventional Distributed RL (DiRL)[Horgan et al., 2018] system. In a DiRL system (Section 5.2), multiple nodes interact with the environment concurrently and their experiences are pooled together in a central server. The server learns from these experiences and broadcasts the updated policies back to the nodes. A naive DiRL implementation does not however imply state-space partitioning or efficient exploration. We propose a novel exploration method, ϵ-pref, to partition the state-space among sensor nodes for efficient exploration. This implicitly results in higher risk of disastrous outcomes for some nodes. We propose ϵ-safe to deal with the trade-off between efficient exploration via state-space partitioning and the risk it entails. Additionally, since each node of the sensor network generally deals with its unique environment, it is important that it adapts to perform optimally as fast as possible. This is possible by decreasing instances of unnecessary exploration. To this end, we present ϵ-adapt to dynamically adjust the exploration rate. Our main contributions in this work are:

- We propose ϵ-pref to partition the state-space for efficient exploration by coordinating the ϵ-greedy behavior in a DiRL system. We achieve this by allotting each node in an EHWSN network with a preferred exploratory action such that each node explores a specific region of the state-space (Section 5.2.4).

- We also propose ϵ-safe, a technique to distribute the risk associated with exploration uniformly among all nodes so as to increase the collective performance without compromising heavily on exploration efficiency (Section 5.2.5).

- Additionally, we propose ϵ-adapt, to automatically adapt the exploration rate so as to minimize unnecessary exploration and decrease the learning time and cost (Section 5.2.6).

We describe our proposed system in Section 5.2. The evaluation setup is explained in Section 5.3 and the results are discussed in Section 5.4.

5.2 Proposed DiRL System and Exploration Strategies

In this section, we describe the distributed ENO-RL system and propose our novel exploration strategies to accelerate learning.

5.2.1 DiRL System

We consider a distributed EHWSN network that consists of one central *learner* (corresponding to a network server) and a fleet of N_w *workers*. Each worker corresponds to an EHWSN which uses the same basic ENO-RL system model described previously in Section 4.2. The EHWSN harvests solar energy to power itself and regulates its power consumption through duty cycling. The duty cycling policy is maintained by the APM. The APM uses DQN to learn policies to optimize the duty cycles of the node. Specifically, we employ Double DQN with Dueling Architectures to carry out the learning process. We further assume that the learner (central server) is not constrained by computational and energy limitations. For the sake of simplicity, we also assume that the communication between the server and the workers is error-free and not affected by issues like noise and congestion.

During the i-th episode, each worker $w_k, k = 1, 2, ...N_w$, interacts with its unique environment according to a policy $\pi(w_k|\theta_i)$ and records its *experience*. An experience for worker k at time t is a tuple $(s_{kt}, a_{kt}, r_{kt}, \hat{s}_{kt})$ that includes information about the current state s_kt, the action taken a_kt, the reward received r_kt and the resulting next state $\hat{s}_kt$. At the end of an episode, each worker uploads its stream of experiences $(s_{k1}, a_{k1}, r_{k1}, \hat{s}_{k1}), ..., (s_{kT}, a_{kT}, r_{kT}, \hat{s}_{kT})$ to a global memory pool $\mathcal{M}$.

The learner randomly gathers a minibatch of experiences $\mathcal{Z}(\mathcal{M})$ from this pool and performs N_l learning steps using the loss function described in Section 3.2.7. The

updated parameters θ_{i+1}, are then broadcast back to the workers. The workers do not execute any learning steps. We make this assumption because EHWSNs are typically too constrained in their computational resources to perform gradient-based learning. All workers receive the same set of parameters from the learner. In addition to the updated parameters, the workers also receive an exploration policy directive from the central server that determines their unique exploratory behavior.

5.2.2 States and Actions

In this framework, we use a continuous state space in this case. For an episode E, the agent's state at time t is $s_t = (b_t, p_t, h_t, f_E)$. b_t represent the battery level, p_t is the ENP of the node, h_t is the harvested energy level and $f_E \in [1, 2...N_F]$ is a rough estimate of the predicted energy harvesting opportunity for episode E. f_E can be easily obtained from the internet, e.g., weather forecast websites. The action space is discrete. The agent's actions a_t correspond to the possible duty cycles $d \in [\ d_{min}, d_{max}]$.

5.2.3 Reward Function

The reward $r_E \in [-1, 1]$ is calculated only at the end of the episode E and all the state-action pairs that were encountered during E are equally credited with that amount. This reduces reward sparsity and stabilizes the learning process. This rewarding scheme is different from the one in the previous chapter (Section 4.3.1). For the SARSA MDP, only the last state-action pair of an episode is associated with a reward (all intermediate state-action pairs receive zero reward) and therefore the rewards were sparse. The credit-assignment to all the contributing state-action pairs was achieved by using the eligibility trace. In the case of continuous state spaces, it is not possible to use eligibility traces to broadcast the rewards back to the contributing state-action pairs. Hence, we wait until the end of an episode to calculate the reward and associate it with all the contributing state-action pairs before storing it into the experience replay buffer. This is valid because the learning step is carried out in the

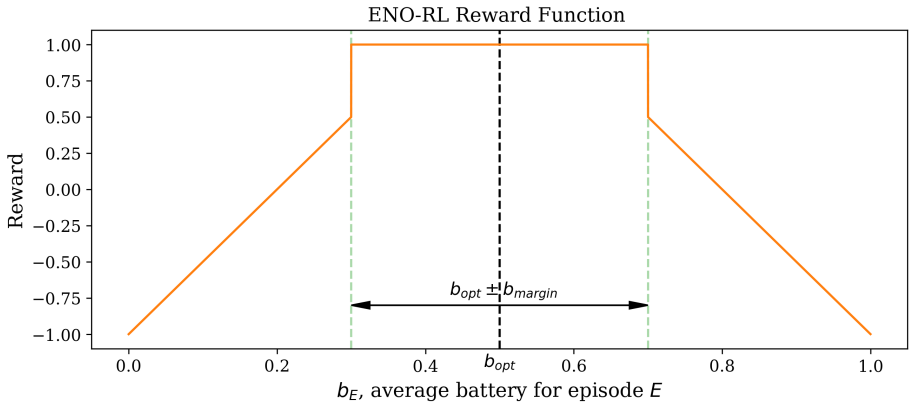

Figure 5-1: The reward for episode E depends upon the mean battery level b_E during that episode. The rewards are clipped at ± 1 to ensure stability during training.

central server and not in the nodes themselves. Hence, we can wait until the the end of the episode before transmitting the experiences to the central server to update the DQN parameters. Although the learner is using slightly state experiences to update the NN parameters, this is not a problem for DQNs because it is an off-policy learning algorithm.

r_E is based on the mean battery level b_E, for episode E (Figure 5-1). We do not shape the reward to prefer any particular value of b_E as long as it is within the working range of $b_{opt} \pm b_{margin}$. The rewards decrease linearly as b_E deviates further from this range. A large b_{margin} implies greater uncertainty about the optimal battery level. While this increases the complexity of the problem, it greatly relaxes the effect of the choice of hyperparameter b_{opt} which is not easily obtainable (b_{opt} is usually obtained from historical data which may not always be available). r_E is given by:

$$r_E = \begin{cases} 1, & \text{if } |b_{opt} - b_E| \leq b_{margin} \\ 1.5 - 5 \times \frac{|b_{opt} - b_E|}{b_{max}}, & \text{otherwise} \end{cases} \tag{5.1}$$

This reward function encourages the agent to maintain the battery levels within a safe working range $b_{opt} \pm b_{margin}$, while allowing the battery levels to fluctuate sufficiently during an episode. The constants used in the Equation (5.1) were determined empirically.

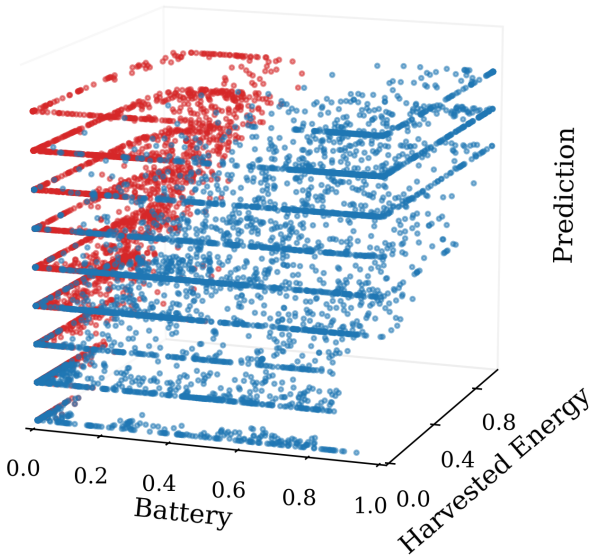

Figure 5-2: A clear state-space partitioning as a result of different exploration policies of two agents, Red (high duty cycles) and Blue (low duty cycles). The discrete tiers correspond to each of the discrete weather prediction states (top one corresponds to a sunny day).

5.2.4 Partitioned ϵ-greedy Exploration: ϵ-pref

Motivating Example: Let us take two RL agents, *Red* and *Blue*, attempting to optimize the duty cycles in the ENO-RL system in a non-distributed framework. Red always chooses d_{max} during ϵ-greedy exploration and Blue always chooses d_{min}. Figure 5-2 shows the combined scatter plot of the state-space (ENP state not shown for clarity) visited by each of the agents after one year of training. Agent Red explores using a high duty cycle and consequently spends more time in regions of low battery levels. Similarly Blue has a lower duty cycle and therefore experiences higher battery levels. From this figure, we observe that it is possible to divide the state-action space between agents by biasing their preferred exploratory actions. Furthermore, we can expect each of these agents to learn from each others' experience to enhance their collective intelligence. We make use of these insights and propose our coordinated exploration methods for the following DiRL Framework.

Let us define a function $\Omega(w_k) = k$ that maps each worker w_k to a unique real

number k. Let $N_A = N_D$ be the total number of possible discrete actions and let $\tilde{a}$ denote the greedy action. For a given node w, the probability of taking an action $a \in \mathcal{A}$ is:

$$p(a|w) = \begin{cases} 1 - \epsilon + \frac{\epsilon}{N_A}, & \text{if } a = \tilde{a} \\ C(a,w)\frac{\epsilon}{N_A}, & \text{otherwise} \end{cases} \tag{5.2}$$

For naive ϵ-greedy method, $C(a, w) = 1$ and all actions are equally probable to be the exploratory action. This acts as our baseline method for comparison between DiRL methods. We refer to it as ϵ-naive method. In this method, the diversity in experiences is assumed to be a consequence of the stochasticity of the unique environments experienced by each worker node. In cases when the stochasticity of the environment is insufficient to induce diverse exploratory behavior, the experiences gathered are redundant. Consequently, while ϵ-naive methods increase the quantity of experiences to learn from, these experience are not necessarily diverse. We describe several methods to overcome this problem below.

The state values corresponding to the battery level (and therefore ENP) are highly correlated with the choice of actions as we saw in Figure 5-2. In contrast, the states for harvested energy and weather prediction are completely independent of the node duty cycles. We can hope to divide the state-space only on the basis of battery level. We do so by dedicating each node with a preferred duty cycle for exploration by a *preference* factor of $C(a, w)$ as follows:

$$C(a,w) = \begin{cases} u + \frac{1-u}{N_w}, & \text{if } a = \Omega(w) \\ \frac{1-u}{N_w}, & \text{otherwise} \end{cases} \tag{5.3}$$

Equations (5.2) and (5.3) describe a coordinated exploratory scheme, referred to as ϵ-pref, where a node w is instructed by the central learner to prefer action $a = \Omega(w)$ for exploration by a probability u over other actions. As a result, even if all the nodes experience the same state, the state-action space is maximally explored because the nodes try out all the different possible actions (assuming $N_A \leq N_w$) which may possibly lead the nodes to different regions of the state-space.

5.2.5 Safe ϵ-greedy Exploration: ϵ-safe

ϵ-pref maximizes state-space exploration but as a result, some nodes always end up taking risky actions and facing disastrous results. This reduces the performance of the nodes and affects the overall system performance. To distribute the risks without compromising exploration, we propose a safe exploration method that we call ϵ-safe, where nodes switch their preferred exploratory action every iteration (or episode). If we redefine the condition for a in (5.3) for the i-th iteration as $a = (\Omega(w_k) + i) \mod N_A$, the preferred actions of all nodes change with every iteration. This way, each node explores an action for one iteration (T timesteps). Doing so allows the node to explore that action sufficiently without incurring too much risk. Furthermore, we also dynamically change the preference of exploration, u, such that $u \propto \epsilon$. This means that when the node is exploring at a high rate, it tends to prefer one particular action for exploration during an episode. In the next episode, it will explore with preference to another action. This way, during exploration, the node cycles through all its actions, trying one preferentially in each episode. As ϵ decreases, the preferences become less pronounced.

5.2.6 Adaptive ϵ-greedy Exploration: ϵ-adapt

In ϵ-greedy methods, like the ones discussed above, a typical strategy is to start with a high value of ϵ and gradually anneal it as training progresses. This ensures high exploration at early stages of training. Naive ϵ annealing methods may result in sub-optimal learning because the agent may start acting greedily before it has explored enough and converge sub-optimally; or it may explore for a longer time than required thus missing out on chances for maximizing utility. Furthermore, one fixed annealing method cannot be expected to be optimal for all environments. We propose an adaptive scheme, referred to as ϵ-adapt, to automatically adapt the rate of exploration during learning based on achieved rewards. The basic idea is to explore more when rewards are low (i.e., the agent is behaving sub-optimally) and to behave progressively greedily as rewards increase. We propose an adaptive method where

positive rewards reduce ϵ by a factor of β and negative rewards increase it by the same factor. Expressed mathematically, if the i-th episode acquires a reward r_i using an exploration rate ϵ_i, the exploration rate for next episode, $\epsilon_{i+1} \in [\epsilon_{min}, \epsilon_{max}]$, is given by:

$$\epsilon_{i+1} = \epsilon_i(1 - \beta\frac{r_i}{|r_i|}) \tag{5.4}$$

This method of exploration results in faster adaptation to the nodes' unique working environments and increase in utility.

5.3 Evaluation Setup

5.3.1 Simulation Environment

We simulate the ENO-RL system using realistic values (Table 5.1) based on a TMote Sky [TMote, 2019]. Like in the previous chapter, we use the solar data from JMA [Japan Meteorological Agency, 2019] to calculate harvested solar energy. Each timestep is one hour long. An episode consists of 24 timesteps and corresponds to one day. We split up the days into $N_F = 10$ categories based on their total solar radiation and use it to simulate the coarse predictor of future energy, f_E.

Every hour, the ENO-RL APM identifies its state s_t and chooses an action a_t corresponding to a duty cycle d_t from ten equally spaced discrete duty cycles ($N_D = 10, d_{min} = 10\%$). At the end of the day (episode), the APM calculates its reward r_E, and uploads the experience into its memory pool. The APM collect experiences for a week before it starts learning, after which learning takes place at every time step.

5.3.2 Training Parameters

The DQN uses the hyperparameters listed in Table 5.2. For DiRL systems, the nodes receive an update of their DQN parameters from the central server along with an ϵ-greedy policy directive. This directive sets the value of ϵ and $C(a, w)$ for each node. The nodes then interact with their respective environments and upload their experiences at the end of 24 hours (one episode). On receiving the experiences from the

Table 5.1: Simulation Parameters

Parameter	Value	Description
T	24	Time steps per episode
b_{max}	10.0 Wh	Maximum battery level
b_{opt}	5.0 Wh	Optimal battery level
b_{margin}	$\pm$3.0 Wh	Maximum deviation from b_{opt}
h_{max}	1.0 Wh	Maximum harvested energy per timestep
z_{avg}	0.5 Wh	Mean node energy consumption per timestep
d_{min}	10%	Minimum duty cycle
d_{max}	100%	Maximum duty cycle
N_D	10	No. of duty cycles
N_F	10	No. of weather forecast levels

different nodes, the central server trains for $N_l = 1000$ iterations and then broadcasts the updated parameters and exploration directives back to the nodes.

We determined the annealing rate for ϵ empirically. In single node non-DiRL systems, we anneal ϵ from 0.9 to 0.01 at a rate of 0.1 per year. For the DiRL systems, ϵ_i for episode (or day) E_i is given by:

$$\epsilon_i = 0.9 - \frac{E_i}{E_i + 40} \tag{5.5}$$

5.3.3 Evaluation Metrics

We compare between different methods on the basis of battery violations (downtimes or overflows) because this is a direct indicator of ENO. We define the *learning time* to be the time required for a policy to achieve ENO. ENO is said to be achieved if the number of violations is less than 24 in a span of 365 consecutive days. We also define the *learning cost* as the number of violations committed by a node before achieving ENO. For DiRL system, we sum the violation hours of *all* the nodes. Ideally we would like to minimize both learning cost and time. To compare the state-space penetration in a given time interval between different methods, we take the covariance $\sigma(b, h)$, of the state-values of battery b, and harvested energy h. This makes sense because the other state variables, ENP and prediction, are dependent on b and h. High values of $\sigma(b, h)$ indicate more state space coverage.

Table 5.2: DQN Hyperparameters

Architecture	Double DQN with Dueling Networks [Mnih et al., 2015, Wang et al., 2015b] [Van Hasselt et al., 2016]

Hyperparameters	**Single Agent RL**	**Distributed RL**
hidden layers	1	
hidden layer width	50	
activation	ReLu	
initialization	Kaiming, Xavier	
minibatch size	32	
replay memory size	12,096	
target update frequency, N_u	241,920	
discount factor	0.999	
loss function	mean squared error	
optimizer	ADAM[Kingma and Ba, 2014]	
no. of learners	1	
no. of workers	1	10
learning rate	0.001	
learning steps per day	24	1000

[1] The values of the hyperparameters were obtained empirically using the ENO-RL environment based on weather data for Tokyo, 2000.

[2] The fully connected layer was initialized using Kaiming Uniform method [He et al., 2015] and the value/advantage layers were initialized using Xavier Uniform method [Glorot and Bengio, 2010]. For added stability, the inputs to the neural network are standardized to have a zero mean and a standard deviation of one.

We want to answer the following questions through our experiments.

- What speedups can we expect to gain from DiRL methods?
- How does partitioning the state-space with ϵ-pref affect learning time and cost?
- Can we minimize the learning costs with ϵ-safe?
- Can an adaptive ϵ-adapt lower learning time and cost?

We compare four different policies for the ENO-RL system, summarized in Table 5.3. All DiRL solutions use ten workers ($N_w = 10$) and one separate learner. We use B-ENO and D-ENO as baselines for comparison. The values of u, β and the annealing rate of ϵ were determined empirically.

Table 5.3: ENO-RL Policies

Policies	**ϵ-greedy Type**	**RL Type**	**Preference, u**
B-ENO	ϵ-naive	non-DiRL	N/A
D-ENO	ϵ-naive	DiRL	0.0
P-ENO	ϵ-pref	DiRL	0.5
S-ENO	ϵ-safe	DiRL	ϵ
A-ENO	ϵ-adapt, $\beta = 0.1$	DiRL	ϵ

Since the nodes of a DiRL system are expected to experience different environments, we simulate this by allowing each worker to interact with an environment based on solar data of Tokyo from different years [1]. B-ENO trains with the solar data starting from 1995. Once a system achieves ENO, we test its robustness by implementing it greedily for a period from 1995 to 2018. We also conduct analytical simulations where all the nodes in DiRL experience identical environments based on solar data for 2000. This is to remove the effect of the environmental stochasticity during exploration for fair analysis of our proposed exploration schemes,

[1] Ideally, we would like to have weather data from the same year but different locations in Tokyo - but due to unavailability of such data, we resort to this scheme.

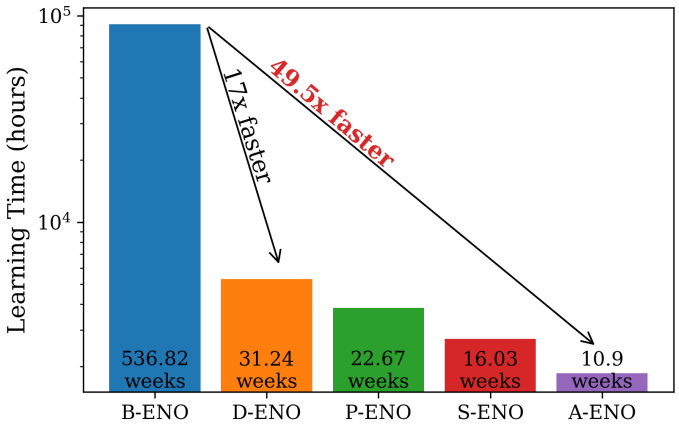

Figure 5-3: A naive DiRL system (D-ENO) accelerates learning by 17x compared to a single agent RL (B-ENO). By coordinating the agents to explore efficiently and using an adaptive exploration rate, learning is accelerated by almost 50x (A-ENO). The numbers in the bars correspond to the time required to achieve ENO.

5.4 Results

In this section, we present the results of our experiments.

5.4.1 Accelerated Learning

Figure 5-3 shows the learning time for the different policies in a log plot. As expected, when we scale up from a single agent to a DiRL system with ten nodes, there is a corresponding dramatic decrease in learning time. With our proposed methods, we are able to achieve even better results and *achieve speedups of up to 49.5x* in the case of A-ENO. This means that with ϵ-adapt methods, an EHWSN network can start performing optimally within 10 weeks of deployment *without any prior training or human intervention.*

Table 5.4: Number of Training and Testing Violations

	B-ENO	**D-ENO**	**P-ENO**	**S-ENO**	**A-ENO**
Training	17722	7930	8817	**5392**	5921
Testing	0	20	16	8	**0**

Table 5.4 lists the violation instances during training and testing of the different

policies. Single node B-ENO has the largest number of violations - longer learning period means more violation instances. D-ENO commits fewer violations due to accelerated learning but we observe that it is not as robust as B-ENO during testing. Both B-ENO and D-ENO use ϵ-naive exploration. The results indicate that simply increasing the number of experiences using DiRL and relying on the stochasticity of the environment to provide diversity may not necessarily result in better policies. Due to inefficient exploration, *D-ENO learns fast but is not optimal.* The poor testing performance of D-ENO likely is the result of overfitting and early convergence to local optima.

5.4.2 State-Space Partitioning with ϵ-pref

P,S,A-ENO coordinate node exploration to explore a wider state-space to overcome the limitations of D-ENO. This is achieved by assigning each node a different preferential exploratory action. As a result, P-ENO fares better in the test compared to D-ENO (Table 5.4); however, more exploration comes with higher learning costs. *P-ENO learns better but costs more.*

To further analyze the effects of coordinated exploration, we look at Figure 5-4 and Figure 5-5 obtained through analytical simulations where all nodes experience identical environments. Figure 5-4 shows the battery profiles for the different policies in the first week of training. In the case of D-ENO, all nodes have very similar battery profiles. However the nodes with P,S,A-ENO policies experience widely different battery levels as a result of coordinated exploration (ϵ-pref, ϵ-safe, ϵ-adapt). This diversity in the state-space exploration is further illustrated in Figure 5-5 as a scatter plot of the states visited. The bottom part of the figure shows the covariance between battery levels and harvested energy, for each day, for the first hundred days. We observe that D-ENO has the least spread (and therefore least covariance) resulting in lesser than par performance. All the other policies cover a wider state-space. A-ENO has the highest variance (four times that of B-ENO) and experiences a wider range of battery levels (downtimes and overflows) in its very first week of training, resulting in reduced learning costs and superior acceleration. As expected, the covariance

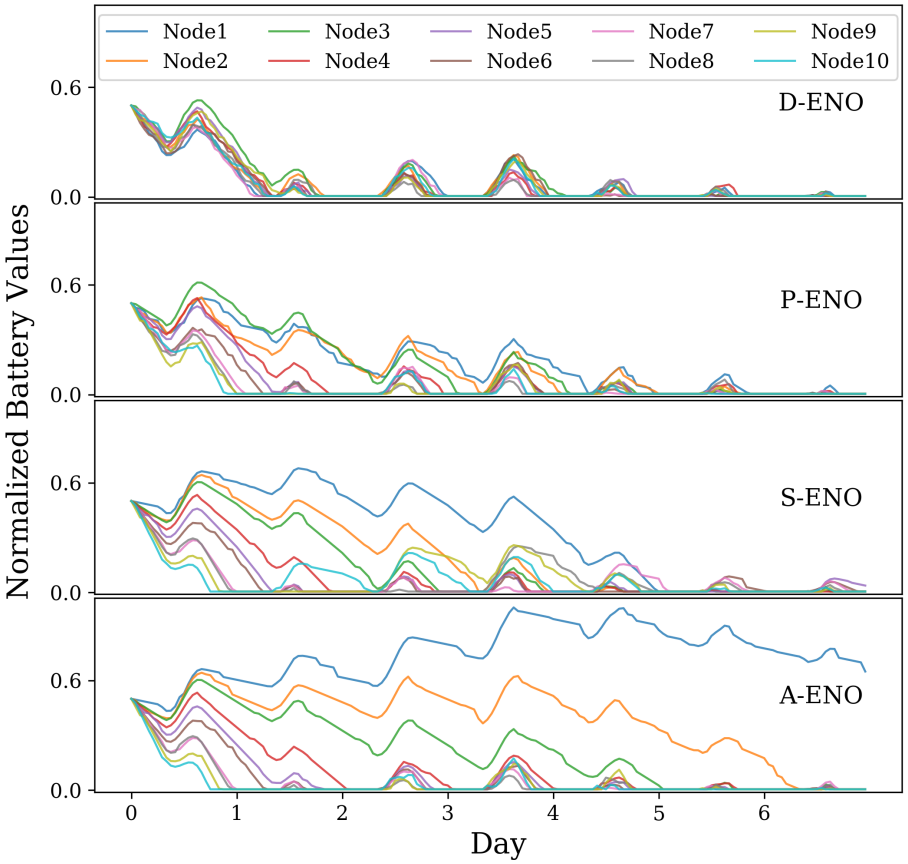

Figure 5-4: Battery profiles for the first week of training. P,S,A-ENO experience diverse battery states compared to D-ENO as a result of coordinated exploration.

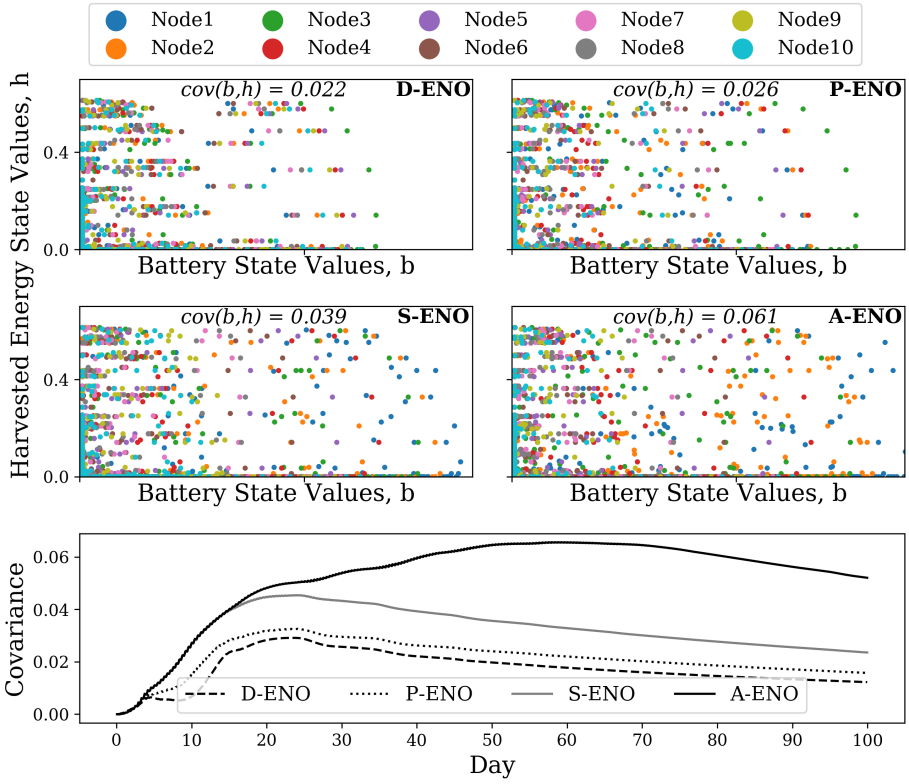

Figure 5-5: Scatter plot of the states visited by different policies in the first two weeks of training. A-ENO has the most spread, i.e., better exploration compared to other methods. A-ENO has the highest covariance between the state values corresponding to battery levels and harvested energy during the first 100 days of training (bottom figure).

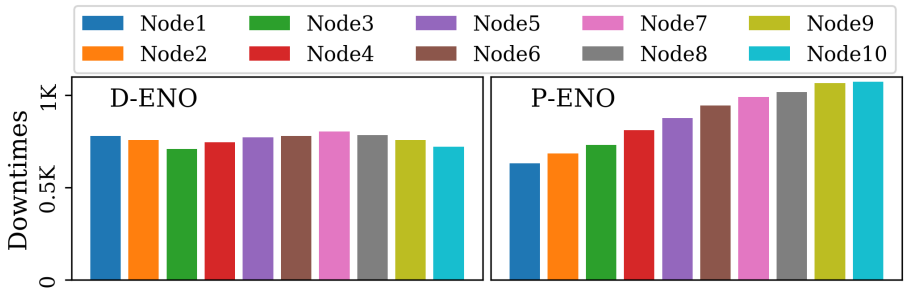

Figure 5-6: The figure shows the cumulative downtimes of all the nodes for D-ENO and P-ENO in the first month. All nodes experience about the same number of downtimes in D-ENO. In contrast, owing to the risks of exploration by partitioning the state-space, the number of downtimes committed by each of the nodes of P-ENO vary significantly (higher duty cycle nodes experience more downtimes).

decreases as exploration rates decrease and the nodes act more greedily towards the end of the 100 day period.

5.4.3 Safe Exploration with ϵ-safe

We now discuss the effects of safe exploration. By assigning each node a different duty cycle with ϵ-pref, some nodes experience more violations than other. For instance, in the battery profiles for P,S,A-ENO in Figure 5-4, Nodes 8-10 drain their batteries quicker than other nodes because they explore using higher duty cycles. Figure 5-6, also obtained through analytical simulations, illustrates this more clearly. It is clear from the figure that some nodes are at more risk of downtimes due to their exploration policy.

While it is inevitable that the nodes suffer through these violations to learn better, we can better distribute these risky situations to reduce learning costs. This is achieved by ϵ-safe in S-ENO. By cycling the exploratory actions with every iteration, the learning costs are reduced: S-ENO has lower costs *and* better results compared to D-ENO and P-ENO as shown in Table 5.4. With safe exploration we can reduce the learning costs by three times compared to B-ENO. This illustrates that it is possible to distribute the risk of exploration without compromising the policy performance.

A comparison of the cumulative learning costs for all DiRL methods is shown in Figure 5-7. A-ENO has a slightly larger learning cost compared to S-ENO, a trade-off

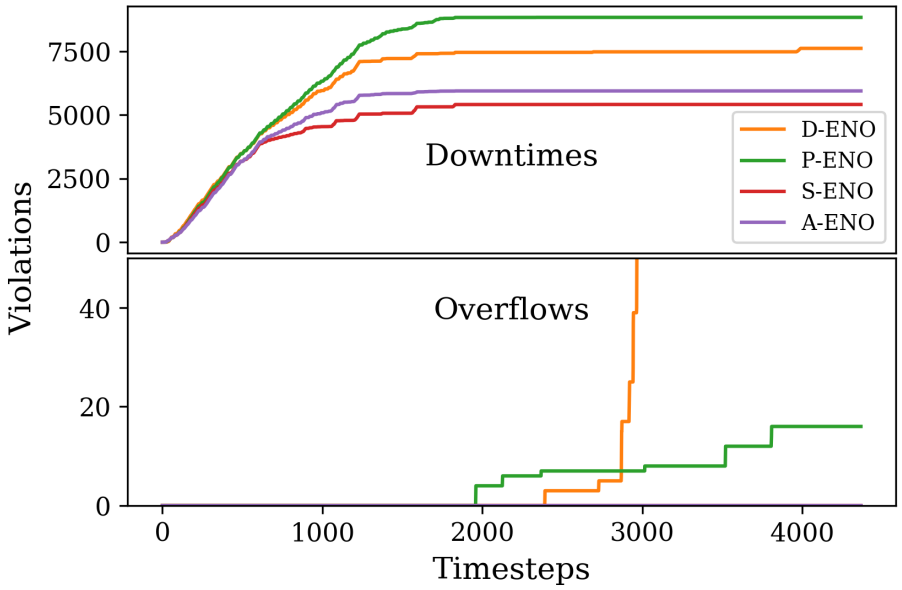

Figure 5-7: The figure shows the number of violations for different policies in the first half of the training year. S-ENO has the lowest number of violations followed by A-ENO due to their cyclic preference of exploratory actions. D-ENO has unstable learning illustrated by the sudden increase in overflows around the 3000^{th} timestep.

for better test results, i.e., *S-ENO trades off safety with robustness.* D-ENO is able to decrease its downtimes when compared to P-ENO but commits many more overflows instead. This indicates improper function approximation due to insufficient variety in training experiences. P-ENO has the highest downtimes because the risks associated with ϵ-pref are unevenly distributed.

In D,P,S-ENO, all the nodes follow the same annealing rate for ϵ. This may not be optimal for nodes to adjust their unique working environment. We get better results by dynamically adapting the exploration rate with ϵ-adapt.

5.4.4 Adaptive Exploration with ϵ-adapt

For A-ENO, the exploration rate depends on the reward received in the previous episode. A positive reward entices the node to act more greedily whereas a negative reward encourages it to explore more. This requires the node to have some minimal amount of exploratory experience before it can start to adapt. Thus in our experiments, A-ENO starts dynamically adapting its exploration rate only after ϵ has annealed to a value below 0.5. Once it starts to adapt ϵ, we limit its value at a

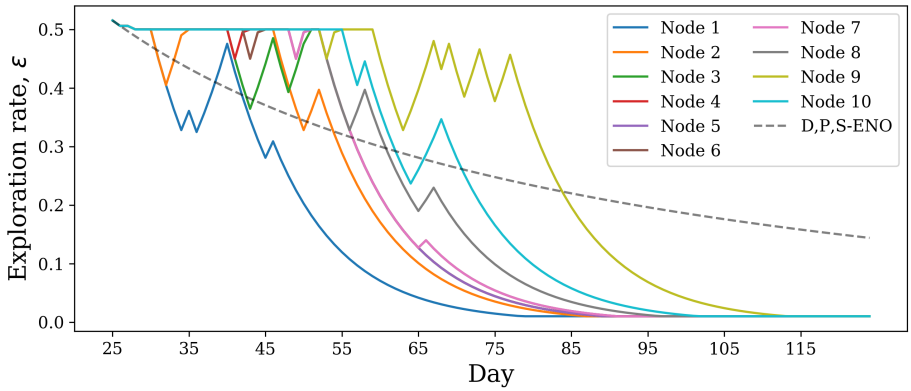

Figure 5-8: The figure shows the exploration rates for different nodes of A-ENO during training. ϵ-adapt encourages greedy behavior if rewards are positive. Nodes that start accumulating rewards quickly anneal their ϵ faster. The dashed gray line shows the non-adaptive ϵ-decay for D,P,S-ENO.

maximum of 0.5. We do this so that excessive loss in rewards due to exploration does not cause the node to act in a complete random manner.

Figure 5-8 shows how ϵ adapts for different nodes of the A-ENO system. The adaptive behavior of the A-ENO system is triggered around the 25^{th} day of training. Node 1 (blue) accumulates rewards faster than its counterparts and quickly anneals its ϵ, i.e., given Node 1's environment, exploration seemed to be redundant. As a result, the node started acting more greedily and accumulating even more rewards.

Node 9, on the other hand, is having difficulty in receiving positive rewards even until the 75^{th} day, probably owing to a difficult environment. Hence, it keeps exploring until the central server learns a good enough policy. In doing so, Node 9 not only increases its rewards but all the other nodes also benefit from its exploratory experiences. By the 115^{th} day, all nodes have started acting mostly greedily. The spikes in their curves correspond to episodes of negative rewards - this may have happened due to exploration or a faulty policy.

Dynamic exploration ensures that nodes do not blindly anneal their ϵ on the basis of time. Rather, the performance of the nodes (indicated by rewards) dictates the annealing rates. Consequently, *A-ENO learns efficiently by adjusting ϵ according to the working environment.* We observe that *A-ENO achieves perfect ENO 50 times faster than B-ENO and with one third of the learning cost* (Figure 5-3 and Table 5.4).

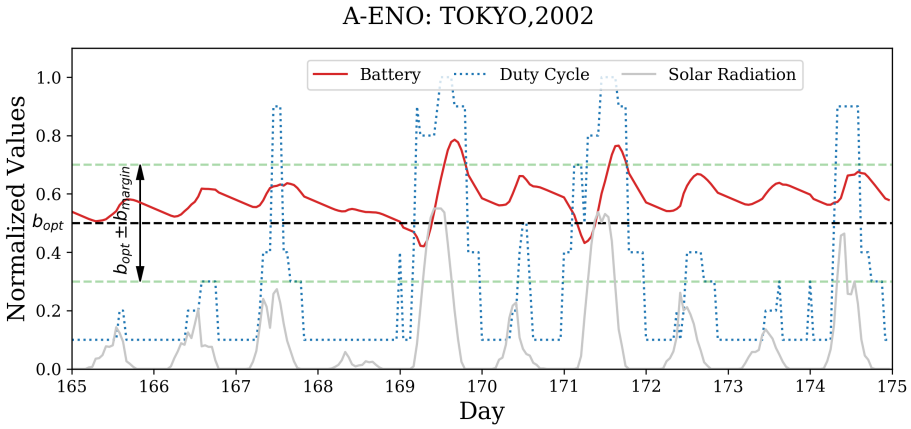

Figure 5-9: Intelligent duty cycling policy learned by A-ENO.

As mentioned before, this superior performance comes with some increase in risky exploratory behavior.

5.4.5 Optimal Duty Cycling for ENO-RL

Figure 5-9 shows the duty cycling behavior corresponding to a greedy implementation of A-ENO policy for 2018 (during testing). It shows the battery and solar profile as well as the duty cycles for ten consecutive days. We observe that the node adapts its duty cycle to the diurnal variations and the widely different solar energy profiles to maintain ENO. The policy has learned that it is optimal to decrease the duty cycles during nighttime and intelligently increase the duty cycles during the day so that the battery level is within $b_{opt} \pm b_{margin}$.

Furthermore, we also observe how the power manager adapts its duty cycle to accommodate the varying energy availability due to seasonal variations in Figure 5-10 and Figure 5-11. Similar to the policy in the earlier chapter (which uses tabular SARSA), the DQN policy is able to maximize duty cycles during summer days with long hours of sunshine and minimize the duty cycles when there is not enough sunshine.

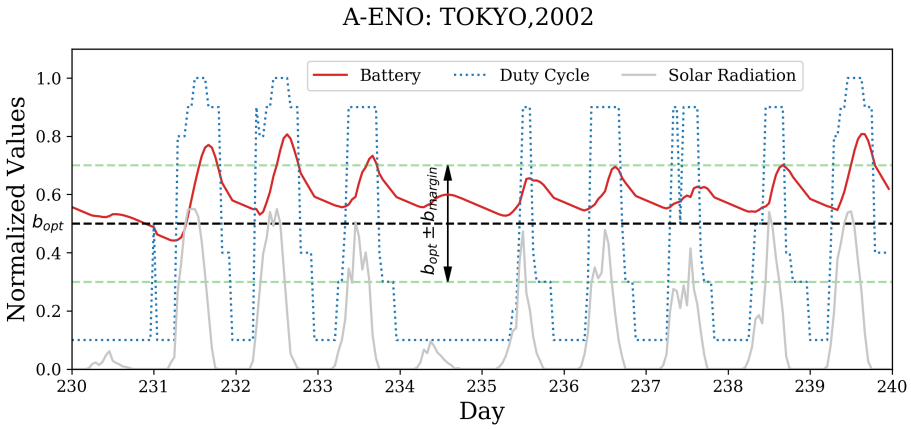

Figure 5-10: Battery profile of ENO-RL using A-ENO for Tokyo, 2002 from 230$^{\text{th}}$ to 240$^{\text{th}}$ day. The battery fluctuates around the 50% mark and hence the node is ENO. The nodes has high duty cycles during sunny periods and low duty cycles during the night and day with low sunshine. Note that the duty cycles never dip below 0.1%. On the 234$^{\text{th}}$ and 235$^{\text{th}}$ day, the solar energy is scarce and the APM intelligently lowers the duty cycle.

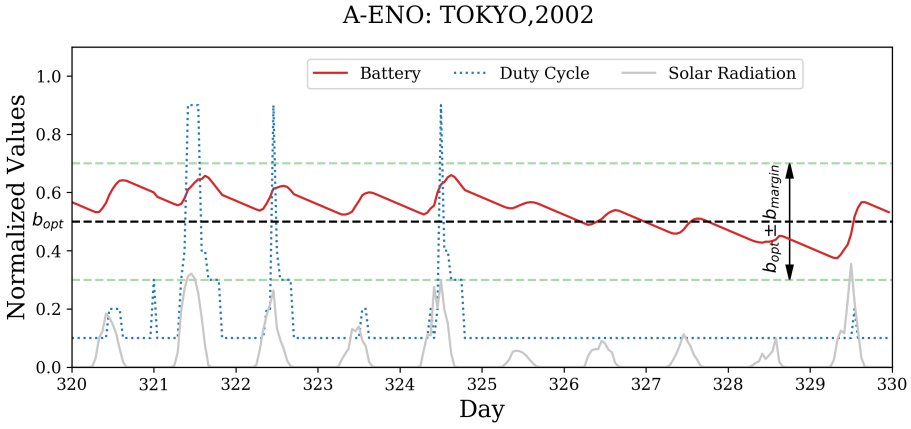

Figure 5-11: Battery profile of ENO-RL using A-ENO for Tokyo, 2002 from 320$^{\text{th}}$ to 330$^{\text{th}}$ day. The APM successfully negotiates consecutive days of low solar energy without any battery violations while maximizing the duty cycle when possible.

5.5 Conclusion

We propose coordinated ϵ-greedy exploration methods to partition the state-space among different agents to explore efficiently during DiRL and accelerate training in ENO-RL systems. To overcome the non-uniform distribution of risk associated with coordinated exploration, we propose methods to trade-off performance and safety. Furthermore, by intelligently adjusting the exploration rates and the preference factors, we decrease instances of unnecessary exploration. Our methods accelerate learning speeds by an order of magnitude and lowers violations during training by upto one thirds. We can conclude that using our methods, it is possible to use a comparatively simple ϵ-greedy exploration to optimize exploration for a large state-space. As a result, EHWSNs can work optimally in an environment that it has never experienced before with less than half a year of training and reduced learning costs.

Chapter 6

Multi-Objective Framework for ENO

Modern Energy Harvesting Wireless Sensor Nodes (EHWSNs) need to intelligently allocate their limited and unreliable energy budget among multiple tasks to ensure long-term uninterrupted operation. Analytic solutions to this multi-objective problem are application-specific and rely heavily on non-causal a priori information. Alternatively, as we have seen in the previous chapters, learning-based approaches have the potential for a unified and adaptive energy scheduling framework. However, they have not been sufficiently explored for multi-objective optimization. These learning-based solutions are unsuitable for multiple objectives and cannot execute a posteriori tradeoffs. In this chapter, we propose a general MORL framework for ENO of EHWSNs. Our proposed framework consists of a novel MOMDP formulation and two novel MORL algorithms. Using our framework, EHWSNs can learn to tradeoff between multiple tasks with low learning costs. The high computation costs, usually associated with powerful MORL algorithms, can be avoided by using our comparatively less resource-intensive MORL algorithms. We evaluate our framework on a general single-task and dual-task EHWSN system model through simulations. With our MOMDP, the learning risks are decreased by up to 70% compared to existing solutions and our MORL algorithms can successfully tradeoff between multiple objectives at runtime.

6.1 Introduction

EHWSNs generally operate in environments with unpredictable energy availability and service demands. It becomes necessary for them to judiciously manage their limited energy resources among various tasks to maximize their utility while ensuring energy neutrality for uninterrupted, long-term operation. ENO requires that the average supply and consumption of energy remain balanced within the constraints of causality i.e., cumulative consumed energy must be less than or equal to harvested energy.

Traditionally, EHWSNs were simpler and it was sufficient to maximize a *single* metric such as the communication throughput [Ozel et al., 2011, Ortiz et al., 2016, Aoudia et al., 2018] or the sensing rate [Vigorito et al., 2007, Dias et al., 2016, Murad et al., 2019] in an energy-aware fashion. Modern EHWSNs are more sophisticated and are typically required to execute a variety of tasks related to sensing, communication and processing [Ma et al., 2019, Nakamura et al., 2017]. In such a case, the node needs to optimally allocate the energy between different tasks to maximize its utility and maintain ENO. Take for example, a general purpose solar EHWSN that continuously monitors some physical value like temperature or location. The node encodes and compresses the sensed data before transmitting it to a server. Sometimes the node will have to vary its sensing rate/radius (or transmission power/processor frequency) depending on the utility of sensed data (or channel noise/application process) [Geissdoerfer et al., 2019, Ortiz et al., 2016, Chong et al., 2011]. Furthermore, in some cases, the user may give a higher priority to sensing or communication processes even at the cost of violating energy neutrality. Alternatively, the user may sometimes insist on uninterrupted node operation by agreeing to compromise on the tasks' performance. There are multiple task objectives that need to be optimized simultaneously, but since all tasks draw energy from the same limited source, tradeoffs need to be made based on user-defined preference or *priority*. It is worth emphasizing that the relative priorities are generally not known beforehand and can be adjusted only *after* observing the possible tradeoffs. It is therefore

necessary to develop an intelligent energy management strategy that ensures ENO and optimizes *a posteriori* tradeoffs between different tasks while complying with the dynamic runtime demands of the user. This naturally points to a multi-objective optimization approach.

There have been many research efforts using multi-objective optimization in wireless sensor nodes [Fei et al., 2016]. However, they have mostly concentrated on *network* optimization (rather than *node* optimization). Their optimization objectives (e.g., coverage [Bai et al., 2008], network lifetime [Zhao and Gurusamy, 2008]) and assumptions do not transfer easily to node-centric solutions. Other analytic methods usually rely on non-causal, a priori information and application-specific parameters to construct a solution. They require significant user bias and grow very complex as more tasks are added. Their limited scope and adaptivity is a serious limitation, even for single-objective scenarios, when we consider the continuing massive growth in the number of EHWSN and the diversity in their working environment.

Reinforcement Learning (RL) based alternatives are promising because they use general frameworks to automatically learn energy policies with minimal user input. This gives nodes complete energy autonomy and they can be massively scaled to be used in a deploy-and-forget fashion in new environments. Standard Single-Objective Reinforcement Learning (SORL) methods have already been proposed to derive near-optimal energy neutral policies [Hsu et al., 2014, Dias et al., 2016, Shresthamali et al., 2017, Shresthamali et al., 2019, Xu et al., 2019, Fraternali et al., 2019, Murad et al., 2019] and maximize communication throughput [Blasco et al., 2013, Ortiz et al., 2016, Aoudia et al., 2018, Rioual et al., 2018] using only causal information. However, current SORL methods are not suitable when multiple objectives are concerned primarily because they are based on learning from a single *scalar* feedback reward signal. When multiple objectives are concerned, they are usually projected onto a scalar [Ferreira et al., 2018, Aoudia et al., 2018, Hsu et al., 2014, Murad et al., 2019]. This approach is not a general solution because dynamic tradeoffs cannot be made and it requires a specialized scalarization function with *prior* knowledge of the priorities. For e.g., in [Ferreira et al., 2018], the authors use different sets of predetermined

weights for reward scalarization depending on the mission objective. This prior knowledge is not always available nor easy to determine. This limits the generality of their approach. Thus, there is a need for a *general* energy management framework that can dynamically optimize the energy usage among *multiple objectives* a posteriori in a realistic and feasible manner. This is the primary motivation of our work.

Of course, the assumptions about generality, uninterrupted operation and environment unpredictability can be relaxed for some application scenarios. For e.g., some nodes may have a fairly constant energy source (e.g., thermoelectric generation) or have predictable user requirements (e.g., periodic sensing and transmission schedules). In such cases, simpler heuristic solutions may suffice. In this work however, we are motivated to develop a unified energy management framework that is agnostic about the type of EHWSN and its working environment. This is an urgent necessity because current trends indicate that a massive deployment (in billions) of a wide variety of EHWSNs in novel environments [Cisco, 2020]. At that scale, it is not practical nor tenable to develop individual optimization solutions for each scenario.

6.1.1 The Case for Multi-Objective Optimization (MOO)

In this section, we discuss the limitations of single-objective methods for ENO and we argue that a multi-objective approach using MORL is superior to traditional methods for modern EHWSNs.

Multiple objectives of modern EHWSNs: Consider the SenStick [Nakamura et al., 2017] universal sensing platform used for a variety of applications like human-activity recognition and indoor localization. Its operation involves activating its multiple sensors, processing the gathered data (using an ARM processor) and transmitting it via Bluetooth. Each of these tasks consume non-trivial amounts of power (shown in Table 6.1) and this needs to considered when deciding the energy budget.

Present solutions for EHWSN energy management assume that one of the tasks (e.g., transmission) overwhelmingly consumes a major percentage of the energy and ignore the energy spent in others. For instance, in [Ozel et al., 2011, Blasco et al.,

2013, Ortiz et al., 2016], the authors focus only on maximizing a communication-based performance metric such as throughput or latency. This method is not suitable for modern EHWSNs like SenStick where the energy spent in sensing and processing is also significant. Alternatively, some solutions lump all of the task objectives together and focus on maximizing the gross device energy consumption (assuming that higher consumption implies higher node-utility) [Kansal et al., 2007, Vigorito et al., 2007, Shresthamali et al., 2017]. This approach is limited because it cannot proportion the energy *among the different tasks* w.r.t. to their priorities and does not leave room for dynamic tradeoffs. With our multi-objective approach, it is possible to include many tasks in the optimization process and tradeoff between them, making them more suitable for modern EHWSNs.

The energy neutrality objective: Even if we assume only a single task (or equivalently consider only the gross device consumption), we are still faced with two conflicting objectives: maximizing node-utility (by increasing device power) and maintaining energy neutrality (by reducing device power). By definition, this is already a multi-objective problem.

Analytic methods deal with the dual objectives of ENO by recasting the problem with a single objective and a set of constraints which typically require predictive mechanisms [Kansal et al., 2007, Geissdoerfer et al., 2019]. Maximization of this objective is accomplished by methods such as convex optimization using linear programming [Kansal et al., 2007], non-linear programming [Peng and Low, 2014, Cionca et al., 2018], search methods [Jia et al., 2019], control systems [Vigorito et al., 2007], dynamic programming [Fu et al., 2006, Lei et al., 2016] among other methods. These solutions are not general because the constraints need to be set beforehand and they rely heavily on non-causal a priori information.

Single-objective RL methods learn with a single scalar reward and therefore can optimize only one aspect of the device's performance - either a *single* task-utility [Blasco et al., 2013, Ortiz et al., 2016, Ortiz Jimenez, 2019, Qiu et al., 2019, Kim et al., 2019, Van Huynh et al., 2019, Long and Büyüköztürk, 2020] or the energy neutrality [Hsu et al., 2014, Shresthamali et al., 2017, Dias et al., 2016, Murad et al.,

2019, Shresthamali et al., 2019, Xu et al., 2018, Xu et al., 2019]. However, we run into difficulties as soon as we have to optimize w.r.t. both objectives. Furthermore, this approach is limited also because it cannot optimize for energy neutrality when there are multiple tasks. The policies that only optimize for energy neutrality require that optimization happens over the gross device consumption. They maximize the energy consumption opportunistically whenever surplus energy is present. These policies do not consider the fact that the utility of a task varies with factors other than energy (like user-requests) and that it would be wiser to spend energy in times of high utility over low utility [Geissdoerfer et al., 2019]. With our MORL framework, the node can learn using multiple reward signals and we can express ENO with its own reward signal. In doing so, we allow the user to adjust the "strictness" of its ENO *during* the execution phase. Each of the task rewards can also be defined so that they represent the actual utility which depends not only on the energy consumed but also other factors.

Limitations of scalarization: In modern EHWSNs, sensing, transmission and computation tasks jointly work together to maximize the ultimate objectives of maximizing the node-utility and maintaining energy neutrality. However, defining this ultimate objective with a single scalar metric is not trivial [Rioual et al., 2018]. One popular approach is to project multiple objectives into a scalar by some linear/non-linear function as in [Hsu et al., 2014, Aoudia et al., 2018, Rioual et al., 2018]. This requires the scalarization function and the weights to be fixed and known a priori, like in [Ferreira et al., 2018], which is difficult and is not always possible. Any modification in weights would restart the learning process and therefore eliminates the possibility of a posteriori tradeoffs w.r.t. runtime user-preferences. Also, the scalarization function itself is generally very complicated and may take many forms depending on the application [Hsu et al., 2014]. Improper formulations may lead to unexpected behavior due to reward hacking [OpenAI, 2020a, DeepMind, 2020, Everitt and Hutter, 2019]. Thus, scalarization methods using RL have very debilitating limitations. A multi-objective approach is more attractive because it can learn from simpler, multiple rewards and optimize tradeoffs at runtime.

Task	Minimum	Maximum
Sensing	0.04 mA	3.7 mA
Communication	2.7 mA	7.5 mA
Processing	3.3 mA	7.4 mA

Table 6.1: Current consumption of various tasks in SenStick.

Multi-Objective RL: Our proposed MORL framework is a general solution which can i) allocate energy between multiple tasks, ii) optimize between gross device energy consumption and long-term energy neutrality and iii) dynamically perform tradeoffs during runtime. The framework also accommodates rewards with different characteristics (e.g., binary/discrete/continuous, sparse/dense). By representing each task with a different reward component, we can dispense with complicated hand-tuned scalarization function and use simple, general and intuitive rewards that accurately reflect the task objective. Furthermore, our framework allows us to optimize each task over its own independent optimization horizon (i.e., different discounting factors).

Direct MORL solutions [Pirotta et al., 2015, Wang and Sebag, 2012] are not feasible for EHWSNs because they are either too computationally intensive (for training/inference) and require impractically high learning costs. This is because the RL policies need to learn a very complex function over a vast exploration space. To overcome this limitation, the algorithms in our MORL framework learn a coverage set of greedy policies that are used to determine the optimal action during the execution phase once the user priorities are given. This allows for dynamic tradeoff at runtime without having to constantly maintain and update a set of different policies for different priorities (not scalable) or having to learn and infer from a complex function that represents the entire Pareto-front (too computationally expensive).

Since our work deals with multi-objective optimization, which is already a complex problem, we have not addressed the issue of task dependencies [Xu et al., 2019] at present. Task dependencies add an extra parameter into optimization process which compounds the problem complexity. As a first step, we have assumed that the different tasks of EHWSNs are independent.

6.2 Censored Content

This is a censored version of the original thesis. The remaining parts of this chapter is under review for publication. Since the review process is double-blind, it is necessary to maintain anonymity. Thus, the remaining portion of this chapter, scheduled to be published by 1-October-2022, has been censored. removed. The full-text version of will be available after this date.

Chapter 7

Conclusion

Energy neutrality of EHWSNs is a necessary requirement for their long-term autonomous operation and this still remains an open problem for the research community. Heuristics and analytical methods have a limited scope of applicability due to their non-adaptive nature and the need for significant design bias. General formulations based on analytic methods are impractical if not impossible. RL is an attractive alternative to traditional methods. With learning-based methods, we can develop a common unified learning framework that generally works for most EHWSNs. With RL, EHWSNs can learn intelligent and adaptive policies for long-term energy neutrality with minimal human supervision.

Learning-based methods for ENO are now transitioning from an interesting research question in labs to more urgent practical applications in the field. This is mainly due to the current widespread use of IoT devices and the scale of their deployment. The core motivation of our work is to develop solutions that make implementing RL methods for ENO in EHWSNs more feasible.

Although RL is very attractive solution for ENO, implementing it in EHWSNs is not trivial. Straightforward RL solutions are typically impractical due to the large learning and computation costs. The constraints on energy, computation resources coupled with the inherent unpredictability of the working environment makes it very difficult to implement RL in EHWSNs.

7.1 Summary

In this work, we tackle this problem of realizing RL solutions for EHWSNs from three different directions. First, we develop a basic system model, define a MDP and implement one of the simplest RL (SARSA(λ) algorithm. Our MDP consists of a general reward function and state definition that facilitates stable learning. Our proposed solution requires very little computation and is easily implementable [Fraternali et al., 2019]. We show that the agent converges to policies that are near-optimal, convergent, and adaptive to changes in weather, climate and node-degradation.

Secondly, we develop an even more powerful RL agent using DQNs. To overcome the large training time associated with DQNs, we use a distributed framework to accelerate learning. We coordinate the many nodes in WSNs to simultaneously explore different regions of the state-space. Our exploration methods are able to effectively decrease the learning costs without compromising heavily on the optimality of the solution. Furthermore, we offload all the compute-heavy *learning* to a central server (or WSN base station) to decrease the computation costs.

Finally, we tackle the problem of energy scheduling in multi-task EHWSNs. Modern EHWSNs need to optimize between sensing, communication, and processing tasks. By using naive single-objective RL and scalarization methods, we get sub-optimal solutions that are incapable of tradeoffs and have very high learning costs. We propose a novel MORL framework consisting of a novel MOMDP and two MORL algorithms. Our novel MOMDP framework facilitates efficient learning so that we can learn very complicated policies within reasonable learning costs. Furthermore, our novel MORL algorithms use minimal computation costs to obtain a posteriori tradeoff policies that are learned with very little learning costs.

Thus, in this work we first show the proof of concept by developing an easily implementable RL solution for ENO. We then also provide different strategies to decrease the learning and computation costs when using NN based DRL. We then also extend the problem to multi-objective optimization and propose novel MORL RL algorithms and framework. As a result, EHWSNs are a step closer to autonomous

and perpetual operation. As we inch closer to full autonomy, we can expect IoT to be more seamlessly integrated into our world and enrich our lives.

7.2 Limitations and Future Directions

Although our work overcomes some of the important challenges of using RL for EHWSNs, practical real-world implementation is still not easily within sight. Of our proposed solutions, tabular SARSA(λ) is the most attractive one to date due to its minimal compute requirements. Unfortunately due to constraints in time and experimental resources, our work lacks the real-world evaluation in more specific application scenarios. However, other researchers have corroborated the validity of our approach. In [Fraternali et al., 2019], the authors develop and evaluate a real-world sensor mote based on our MDP (but use tabular Q-learning) which shows that it is a feasible alternative to traditional methods.

While tabular methods lend themselves easily to real-world applications, unfortunately there is still work to be done when realizing DQNs in resource-constrained sensor nodes with a small form factor. For instance, in the DiRL system proposed in Chapter 5, although the sensor nodes use NNs only for inference, the NNs are still a bit too large for the on-board memory. One alternative is to prune the networks to a more realistic size. This is a promising alternative and there has been some interesting research in this area [Iandola et al., 2016, Lin et al., 2020]. From the hardware point of view, encouraging results have been reported in [Restuccia and Melodia, 2020] where the authors implement DRL to optimize the modulation scheme for software-defined radio in real-time low-power hardware. Thus, with intelligent hardware design and compressed NN architectures, it may soon be possible to implement sophisticated DRL solutions for EHWSNs and therefore enabling a high degree of autonomy.

Another point of contention may be the learning costs associated with RL. Although our proposed methods have reduced the learning time and costs dramatically compared to naive RL solutions, they may still be prohibitively too length for realistic scenarios. We would rather if the nodes could learn in a matter of days rather

than years. One approach to accelerate the learning process would be to use many more nodes in the DiRL framework. However, this may lead to instability in DRL due to what is knows as the deadly triad [Van Hasselt et al., 2018] and may require stronger off-policy corrections [Fujimoto et al., 2019, Kumar et al., 2019]. Another methods would be to decrease the granularity of the timesteps in the MDP. In all of our work, we assume each timestep corresponds to one hour (to remain consistent with the solar data [Japan Meteorological Agency, 2019]). However if one were to reduce this, to say, one second, and adjust the discounting factor accordingly, this would dramatically reduce the learning period. However, this would also probably require a larger experience replay buffer and a more intelligent replay policies like Prioritized Experience Replay ([Schaul et al., 2015, Horgan et al., 2018].

On the other hand, MORL is still relatively underdeveloped compared to DRL methods. Current MORL strategies are still undergoing research and are not as robust as their single-objective counterparts. Furthermore, the complexity of designing a MORL framework is compounded by the problem of hyperparameter tuning in RL and NNs. The ML research community has yet to fully understand how and why the different learning hyperparameters affect the learning process and the end results. A deeper insight into the 'psychology' of RL agents and the 'brain circuitry' of NNs would make it much easier to develop more robust solutions - especially w.r.t. MORL methods and novel MORL algorithms. An orthogonal approach for consolidating multiple objectives may be a game-theoretic/multi-agent approach as explored in [Sharma et al., 2019] and [Ortiz et al., 2017]. While this is an interesting approach with a more general problem statement, it adds an additional layer of complexity of social dynamics and interaction.

Bibliography

[Achiam and Sastry, 2017] Achiam, J. and Sastry, S. (2017). Surprise-based intrinsic motivation for deep reinforcement learning. *arXiv preprint arXiv:1703.01732*.

[Al-Karaki and Kamal, 2004] Al-Karaki, J. N. and Kamal, A. E. (2004). Routing techniques in wireless sensor networks: a survey. *IEEE wireless communications*, 11(6):6–28.

[Alippi et al., 2009] Alippi, C., Anastasi, G., Di Francesco, M., and Roveri, M. (2009). Energy management in wireless sensor networks with energy-hungry sensors. *IEEE Instrumentation & Measurement Magazine*, 12(2).

[Alippi et al., 2011] Alippi, C., Camplani, R., Galperti, C., and Roveri, M. (2011). A robust, adaptive, solar-powered wsn framework for aquatic environmental monitoring. *IEEE Sensors Journal*, 11(1):45–55.

[Alsheikh et al., 2014] Alsheikh, M. A., Lin, S., Niyato, D., and Tan, H.-P. (2014). Machine learning in wireless sensor networks: Algorithms, strategies, and applications. *IEEE Communications Surveys & Tutorials*, 16(4):1996–2018.

[Aoudia et al., 2018] Aoudia, F. A., Gautier, M., and Berder, O. (2018). Rlman: an energy manager based on reinforcement learning for energy harvesting wireless sensor networks. *IEEE Transactions on Green Communications and Networking*, 2(2):408–417.

[Ashton et al., 2009] Ashton, K. et al. (2009). That 'internet of things' thing. *RFID journal*, 22(7):97–114.

[Bai et al., 2008] Bai, X., Xuan, D., Yun, Z., Lai, T. H., and Jia, W. (2008). Complete optimal deployment patterns for full-coverage and k-connectivity (k$<$ 6) wireless sensor networks. In *Proceedings of the 9th ACM international symposium on Mobile ad hoc networking and computing*, pages 401–410.

[Bellman, 1954] Bellman, R. (1954). The theory of dynamic programming. Technical report, Rand corp santa monica ca.

[Benini et al., 2001] Benini, L., Castelli, G., Macii, A., and Scarsi, R. (2001). Battery-driven dynamic power management. *IEEE Design & Test of Computers*, 18(2):53–60.

[Blasco and Gündüz, 2015] Blasco, P. and Gündüz, D. (2015). Multi-access communications with energy harvesting: A multi-armed bandit model and the optimality of the myopic policy. *IEEE Journal on Selected Areas in Communications*, 33(3):585–597.

[Blasco et al., 2013] Blasco, P., Gunduz, D., and Dohler, M. (2013). A learning theoretic approach to energy harvesting communication system optimization. *IEEE Transactions on Wireless Communications*, 12(4):1872–1882.

[Braten and Kraemer, 2018] Braten, A. E. and Kraemer, F. A. (2018). Towards cognitive iot: Autonomous prediction model selection for solar-powered nodes. In *2018 IEEE International Congress on Internet of Things (ICIOT)*, pages 118–125. IEEE.

[Buchli et al., 2014] Buchli, B., Sutton, F., Beutel, J., and Thiele, L. (2014). Dynamic power management for long-term energy neutral operation of solar energy harvesting systems. In *Proceedings of the 12th ACM conference on embedded network sensor systems*, pages 31–45.

[Capella et al., 2013] Capella, J. V., Bonastre, A., Ors, R., and Peris, M. (2013). In line river monitoring of nitrate concentration by means of a wireless sensor network with energy harvesting. *Sensors and Actuators B: Chemical*, 177:419–427.

[Chan et al., 2015] Chan, W. H. R. et al. (2015). Adaptive duty cycling in sensor networks with energy harvesting using continuous-time markov chain and fluid models. *IEEE Journal on Selected Areas in Communications*, 33(12):2687–2700.

[Chebrolu et al., 2008] Chebrolu, K., Raman, B., Mishra, N., Valiveti, P. K., and Kumar, R. (2008). Brimon: a sensor network system for railway bridge monitoring. In *Proceedings of the 6th international conference on Mobile systems, applications, and services*, pages 2–14.

[Cho et al., 2010] Cho, S., Jo, H., Jang, S., Park, J., Jung, H.-J., Yun, C.-B., Spencer Jr, B. F., and Seo, J.-W. (2010). Structural health monitoring of a cable-stayed bridge using wireless smart sensor technology: data analyses. *Smart Structures and Systems*, 6(5-6):461–480.

[Chong et al., 2011] Chong, S. K., Gaber, M. M., Krishnaswamy, S., and Loke, S. W. (2011). Energy-aware data processing techniques for wireless sensor networks: a review. In *Transactions on large-scale data-and knowledge-centered systems III*, pages 117–137. Springer.

[Cionca et al., 2018] Cionca, V., McGibney, A., and Rea, S. (2018). Mallec: Fast and optimal scheduling of energy consumption for energy harvesting devices. *IEEE Internet of Things Journal*, 5(6):5132–5140.

[Cisco, 2020] Cisco (2020). Cisco VNI Complete Forecast Highlights 2020. Technical report.

[Deb, 2001] Deb, K. (2001). *Multi-objective optimization using evolutionary algorithms*, volume 16. John Wiley & Sons.

[Deb et al., 2002] Deb, K., Pratap, A., Agarwal, S., and Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE transactions on evolutionary computation*, 6(2):182–197.

[DeepMind, 2020] DeepMind (2020). Designing agent incentives to avoid reward tampering. [Online; accessed 4-July-2020].

[Di Mauro and Dragoni, 2015] Di Mauro, A. and Dragoni, N. (2015). Adaptive multipath key reinforcement for energy harvesting wireless sensor networks. *Procedia Computer Science*, 63:48–55.

[Dias et al., 2016] Dias, G. M., Nurchis, M., and Bellalta, B. (2016). Adapting sampling interval of sensor networks using on-line reinforcement learning. In *2016 IEEE 3rd World Forum on Internet of Things (WF-IoT)*, pages 460–465. IEEE.

[Dinh et al., 2016] Dinh, T. D. P., Le, T. N., and Duc, A. V. D. (2016). Toward a global power manager in energy harvesting wireless sensor networks. In *Advanced Computing and Applications (ACOMP), 2016 International Conference on*, pages 90–96. IEEE.

[Ermon et al., 2012] Ermon, S., Xue, Y., Gomes, C., and Selman, B. (2012). Learning policies for battery usage optimization in electric vehicles. *Machine learning and knowledge discovery in databases*, pages 195–210.

[Evans, 2011] Evans, D. (2011). The internet of things: How the next evolution of the internet is changing everything. *CISCO white paper*, 1(2011):1–11.

[Everitt and Hutter, 2019] Everitt, T. and Hutter, M. (2019). Reward tampering problems and solutions in reinforcement learning: A causal influence diagram perspective. *CoRR*, abs/1908.04734.

[Fei et al., 2016] Fei, Z., Li, B., Yang, S., Xing, C., Chen, H., and Hanzo, L. (2016). A survey of multi-objective optimization in wireless sensor networks: Metrics, algorithms, and open problems. *IEEE Communications Surveys & Tutorials*, 19(1):550–586.

[Ferreira et al., 2018] Ferreira, P. V. R., Paffenroth, R., Wyglinski, A. M., Hackett, T. M., Bilén, S. G., Reinhart, R. C., and Mortensen, D. J. (2018). Multiobjective reinforcement learning for cognitive satellite communications using deep neural network ensembles. *IEEE Journal on Selected Areas in Communications*, 36(5):1030–1041.

[Fortunato et al., 2017] Fortunato, M., Azar, M. G., Piot, B., Menick, J., Osband, I., Graves, A., Mnih, V., Munos, R., Hassabis, D., Pietquin, O., et al. (2017). Noisy networks for exploration. *arXiv preprint arXiv:1706.10295*.

[Fraternali et al., 2019] Fraternali, F., Balaji, B., Agarwal, Y., and Gupta, R. K. (2019). Aces–automatic configuration of energy harvesting sensors with reinforcement learning. *arXiv preprint arXiv:1909.01968*.

[Fu et al., 2006] Fu, A., Modiano, E., and Tsitsiklis, J. N. (2006). Optimal transmission scheduling over a fading channel with energy and deadline constraints. *IEEE Transactions on Wireless Communications*, 5(3):630–641.

[Fujimoto et al., 2019] Fujimoto, S., Meger, D., and Precup, D. (2019). Off-policy deep reinforcement learning without exploration. In *International Conference on Machine Learning*, pages 2052–2062. PMLR.

[Gai and Qiu, 2018] Gai, K. and Qiu, M. (2018). Optimal resource allocation using reinforcement learning for iot content-centric services. *Applied Soft Computing*, 70:12–21.

[Geissdoerfer et al., 2019] Geissdoerfer, K., Jurdak, R., Kusy, B., and Zimmerling, M. (2019). Getting more out of energy-harvesting systems: energy management under time-varying utility with preact. In *Proceedings of the 18th International Conference on Information Processing in Sensor Networks*, pages 109–120. IEEE.

[GhasemAghaei et al., 2007] GhasemAghaei, R., Rahman, M. A., Gueaieb, W., and El Saddik, A. (2007). Ant colony-based reinforcement learning algorithm for routing in wireless sensor networks. In *Instrumentation and Measurement Technology Conference Proceedings, 2007. IMTC 2007. IEEE*, pages 1–6. IEEE.

[Glorot and Bengio, 2010] Glorot, X. and Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256.

[Gregori and Gómez-Vilardebò, 2016] Gregori, M. and Gómez-Vilardebò, J. (2016). Online learning algorithms for wireless energy harvesting nodes. In *Communications (ICC), 2016 IEEE International Conference on*, pages 1–6. IEEE.

[Gunduz et al., 2014] Gunduz, D., Stamatiou, K., Michelusi, N., and Zorzi, M. (2014). Designing intelligent energy harvesting communication systems. *IEEE Communications Magazine*, 52(1):210–216.

[Hasselt, 2010] Hasselt, H. (2010). Double q-learning. *Advances in neural information processing systems*, 23:2613–2621.

[He et al., 2015] He, K., Zhang, X., Ren, S., and Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034.

[Hessel et al., 2018] Hessel, M., Modayil, J., Van Hasselt, H., Schaul, T., Ostrovski, G., Dabney, W., Horgan, D., Piot, B., Azar, M., and Silver, D. (2018). Rainbow: Combining improvements in deep reinforcement learning. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

[Horgan et al., 2018] Horgan, D., Quan, J., Budden, D., Barth-Maron, G., Hessel, M., Van Hasselt, H., and Silver, D. (2018). Distributed prioritized experience replay. *arXiv preprint arXiv:1803.00933*.

[Houthooft et al., 2016] Houthooft, R., Chen, X., Duan, Y., Schulman, J., De Turck, F., and Abbeel, P. (2016). Vime: Variational information maximizing exploration. In *Advances in Neural Information Processing Systems*, pages 1109–1117.

[Howard et al., 2002a] Howard, A., Matarić, M. J., and Sukhatme, G. S. (2002a). An incremental self-deployment algorithm for mobile sensor networks. *Autonomous Robots*, 13(2):113–126.

[Howard et al., 2002b] Howard, A., Mataric, M. J., and Sukhatme, G. S. (2002b). Mobile sensor network deployment using potential fields: A distributed, scalable solution to the area coverage problem. *Distributed autonomous robotic systems*, 5:299–308.

[Hsu et al., 2006] Hsu, J. et al. (2006). Adaptive duty cycling for energy harvesting systems. In *Proc. of the 2006 ISLPED*, pages 180–185.

[Hsu et al., 2014] Hsu, R. C., Liu, C.-T., and Wang, H.-L. (2014). A reinforcement learning-based tod provisioning dynamic power management for sustainable operation of energy harvesting wireless sensor node. *IEEE Transactions on Emerging Topics in Computing*, 2(2):181–191.

[Huang and Tseng, 2005] Huang, C.-F. and Tseng, Y.-C. (2005). The coverage problem in a wireless sensor network. *Mobile Networks and Applications*, 10(4):519–528.

[Iandola et al., 2016] Iandola, F. N., Han, S., Moskewicz, M. W., Ashraf, K., Dally, W. J., and Keutzer, K. (2016). Squeezenet: Alexnet-level accuracy with 50x fewer parameters and< 0.5 mb model size. *arXiv preprint arXiv:1602.07360*.

[Jagannath et al., 2019] Jagannath, J., Polosky, N., Jagannath, A., Restuccia, F., and Melodia, T. (2019). Machine learning for wireless communications in the internet of things: A comprehensive survey. *Ad Hoc Networks*, 93:101913.

[Japan Meteorological Agency, 2019] Japan Meteorological Agency (2019). Japan meteorological agency. [Online; accessed 6-July-2019].

[Jia et al., 2009] Jia, J., Chen, J., Chang, G., Wen, Y., and Song, J. (2009). Multi-objective optimization for coverage control in wireless sensor network with adjustable sensing radius. *Computers & Mathematics with Applications*, 57(11-12):1767–1775.

[Jia et al., 2019] Jia, R., Zhang, J., Liu, X.-Y., Liu, P., Fu, L., and Wang, X. (2019). Optimal rate control for energy-harvesting systems with random data and energy arrivals. *ACM Transactions on Sensor Networks (TOSN)*, 15(1):13.

[Jiang et al., 2005] Jiang, X., Polastre, J., and Culler, D. (2005). Perpetual environmentally powered sensor networks. In *Proceedings of the 4th international symposium on Information processing in sensor networks*, page 65. IEEE Press.

[Kalantari and Shayman, 2004] Kalantari, M. and Shayman, M. (2004). Energy efficient routing in wireless sensor networks. In *Proc. Conference on Information Sciences and Systems*.

[Kansal et al., 2007] Kansal, A., Hsu, J., Zahedi, S., and Srivastava, M. B. (2007). Power management in energy harvesting sensor networks. *ACM Transactions on Embedded Computing Systems (TECS)*, 6(4):32.

[Kansal et al., 2004] Kansal, A., Potter, D., and Srivastava, M. B. (2004). Performance aware tasking for environmentally powered sensor networks. *ACM SIGMETRICS Performance Evaluation Review*, 32(1):223–234.

[Khadka et al., 2019] Khadka, S., Majumdar, S., Miret, S., Tumer, E., Nassar, T., Dwiel, Z., Liu, Y., and Tumer, K. (2019). Collaborative evolutionary reinforcement learning. *arXiv preprint arXiv:1905.00976*.

[Khadka and Tumer, 2018] Khadka, S. and Tumer, K. (2018). Evolution-guided policy gradient in reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 1188–1200.

[Khan et al., 2015] Khan, J. A. et al. (2015). Energy management in wireless sensor networks: a survey. *Computers & Electrical Engineering*, 41:159–176.

[Kim and Chung, 2006] Kim, D.-S. and Chung, Y.-J. (2006). Self-organization routing protocol supporting mobile nodes for wireless sensor network. In *Computer and Computational Sciences, 2006. IMSCCS'06. First International Multi-Symposiums on*, volume 2, pages 622–626. IEEE.

[Kim et al., 2019] Kim, H., Shin, W., Yang, H., Lee, N., and Lee, J. (2019). Rate maximization with reinforcement learning for time-varying energy harvesting broadcast channels. In *2019 IEEE Global Communications Conference (GLOBECOM)*, pages 1–6. IEEE.

[Kingma and Ba, 2014] Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

[Kinney et al., 2003] Kinney, P. et al. (2003). Zigbee technology: Wireless control that simply works. In *Communications design conference*, volume 2, pages 1–7.

[Krizhevsky et al., 2012] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.

[Kumar et al., 2019] Kumar, A., Fu, J., Tucker, G., and Levine, S. (2019). Stabilizing off-policy q-learning via bootstrapping error reduction. *arXiv preprint arXiv:1906.00949*.

[Lee and Powell, 2012] Lee, D. and Powell, W. B. (2012). An intelligent battery controller using bias-corrected q-learning. In *AAAI*.

[Lee et al., 2006] Lee, R.-G., Chen, K.-C., Chiang, S.-S., Lai, C.-C., Liu, H.-S., and Wei, M.-S. (2006). A backup routing with wireless sensor network for bridge monitoring system. In *4th Annual Communication Networks and Services Research Conference (CNSR'06)*, pages 5–pp. IEEE.

[Lei et al., 2016] Lei, L., Kuang, Y., Shen, X. S., Yang, K., Qiao, J., and Zhong, Z. (2016). Optimal reliability in energy harvesting industrial wireless sensor networks. *IEEE Transactions on Wireless Communications*, 15(8):5399–5413.

[Libelium, 2021] Libelium (2021). Waspmote-The sensor platform to develop IoT projects. `https://www.libelium.com/iot-products/waspmote/`. [Online; accessed 01/22/2021].

[Lillicrap et al., 2015] Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. (2015). Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*.

[Lin et al., 2020] Lin, J., Chen, W.-M., Lin, Y., Cohn, J., Gan, C., and Han, S. (2020). Mcunet: Tiny deep learning on iot devices. *arXiv preprint arXiv:2007.10319*.

[Lin, 1993] Lin, L.-J. (1993). Reinforcement learning for robots using neural networks. Technical report, Carnegie-Mellon Univ Pittsburgh PA School of Computer Science.

[Liu et al., 2014] Liu, C., Xu, X., and Hu, D. (2014). Multiobjective reinforcement learning: A comprehensive overview. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 45(3):385–398.

[Liu and Elhanany, 2006] Liu, Z. and Elhanany, I. (2006). Rl-mac: A qos-aware reinforcement learning based mac protocol for wireless sensor networks. In *2006 IEEE International Conference on Networking, Sensing and Control*, pages 768–773. IEEE.

[Long and Büyüköztürk, 2020] Long, J. and Büyüköztürk, O. (2020). Collaborative duty cycling strategies in energy harvesting sensor networks. *Computer-Aided Civil and Infrastructure Engineering*, 35(6):534–548.

[Lu et al., 2015] Lu, X., Wang, P., Niyato, D., Kim, D. I., and Han, Z. (2015). Wireless networks with rf energy harvesting: A contemporary survey. *IEEE Communications Surveys & Tutorials*, 17(2):757–789.

[Luong et al., 2019] Luong, N. C., Hoang, D. T., Gong, S., Niyato, D., Wang, P., Liang, Y.-C., and Kim, D. I. (2019). Applications of deep reinforcement learning in communications and networking: A survey. *IEEE Communications Surveys & Tutorials*, 21(4):3133–3174.

[Ma et al., 2019] Ma, D., Lan, G., Hassan, M., Hu, W., and Das, S. K. (2019). Sensing, computing, and communications for energy harvesting iots: A survey. *IEEE Communications Surveys & Tutorials*, 22(2):1222–1250.

[Mainwaring et al., 2002] Mainwaring, A., Culler, D., Polastre, J., Szewczyk, R., and Anderson, J. (2002). Wireless sensor networks for habitat monitoring. In *Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications*, pages 88–97.

[Malaver et al., 2015] Malaver, A., Motta, N., Corke, P., and Gonzalez, F. (2015). Development and integration of a solar powered unmanned aerial vehicle and a wireless sensor network to monitor greenhouse gases. *Sensors*, 15(2):4072–4096.

[Mammeri, 2019] Mammeri, Z. (2019). Reinforcement learning based routing in networks: Review and classification of approaches. *IEEE Access*, 7:55916–55950.

[Mao et al., 2014] Mao, S., Cheung, M. H., and Wong, V. W. (2014). Joint energy allocation for sensing and transmission in rechargeable wireless sensor networks. *IEEE Transactions on Vehicular Technology*, 63(6):2862–2875.

[Martin, 2014] Martin, G. (2014). How the internet of things is more like the industrial revolution than the digital revolution.

[Masadeh et al., 2018] Masadeh, A., Wang, Z., and Kamal, A. E. (2018). Reinforcement learning exploration algorithms for energy harvesting communications systems. In *2018 IEEE International Conference on Communications (ICC)*, pages 1–6. IEEE.

[Michelusi et al., 2013] Michelusi, N. et al. (2013). Energy management policies for harvesting-based wireless sensor devices with battery degradation. *IEEE Tr. on Communications*, 61(12):4934–4947.

[Mihaylov et al., 2009] Mihaylov, M., Tuyls, K., and Nowé, A. (2009). Decentralized learning in wireless sensor networks. In *International Workshop on Adaptive and Learning Agents*, pages 60–73. Springer.

[Mills, 2007] Mills, K. L. (2007). A brief survey of self-organization in wireless sensor networks. *Wireless Communications and Mobile Computing*, 7(7):823–834.

[Min et al., 2000] Min, R., Bhardwaj, M., Cho, S.-H., Sinha, A., Shih, E., Wang, A., and Chandrakasan, A. (2000). An architecture for a power-aware distributed microsensor node. In *Signal Processing Systems, 2000. SiPS 2000. 2000 IEEE Workshop on*, pages 581–590. IEEE.

[Mnih et al., 2016] Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D., and Kavukcuoglu, K. (2016). Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937.

[Mnih et al., 2013] Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. (2013). Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*.

[Mnih et al., 2015] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. (2015). Human-level control through deep reinforcement learning. *Nature*, 518(7540):529.

[Moffaert and Nowé, 2014] Moffaert, K. V. and Nowé, A. (2014). Multi-objective reinforcement learning using sets of pareto dominating policies. *Journal of Machine Learning Research*, 15(107):3663–3692.

[Munos, 2006] Munos, R. (2006). Policy gradient in continuous time. *Journal of Machine Learning Research*, 7:771–791.

[Murad et al., 2019] Murad, A., Kraemer, F. A., Bach, K., and Taylor, G. (2019). Autonomous management of energy-harvesting iot nodes using deep reinforcement learning. *arXiv preprint arXiv:1905.04181*.

[Nair et al., 2015] Nair, A., Srinivasan, P., Blackwell, S., Alcicek, C., Fearon, R., De Maria, A., Panneershelvam, V., Suleyman, M., Beattie, C., Petersen, S., et al. (2015). Massively parallel methods for deep reinforcement learning. *arXiv preprint arXiv:1507.04296*.

[Nakamura et al., 2017] Nakamura, Y., Arakawa, Y., Kanehira, T., Fujiwara, M., and Yasumoto, K. (2017). Senstick: Comprehensive sensing platform with an ultra tiny all-in-one sensor board for iot research. *Journal of Sensors*, 2017.

[Ngai and Yung, 2011] Ngai, D. C. K. and Yung, N. H. C. (2011). A multiple-goal reinforcement learning method for complex vehicle overtaking maneuvers. *IEEE Transactions on Intelligent Transportation Systems*, 12(2):509–522.

[OpenAI, 2020a] OpenAI (2020a). Faulty reward functions. [Online; accessed 4-July-2020].

[OpenAI, 2020b] OpenAI (2020b). Openai spinning up. [Online; accessed 4-July-2020].

[Ortiz et al., 2016] Ortiz, A., Al-Shatri, H., Li, X., Weber, T., and Klein, A. (2016). Reinforcement learning for energy harvesting point-to-point communications. In *2016 IEEE International Conference on Communications (ICC)*, pages 1–6. IEEE.

[Ortiz et al., 2017] Ortiz, A., Al-Shatri, H., Weber, T., and Klein, A. (2017). Multi-agent reinforcement learning for energy harvesting two-hop communications with a partially observable state. *arXiv preprint arXiv:1702.06185*.

[Ortiz Jimenez, 2019] Ortiz Jimenez, A. P. (2019). *Optimization and Learning Approaches for Energy Harvesting Wireless Communication Systems*. PhD thesis, Technische Universität.

[Ozel et al., 2011] Ozel, O., Tutuncuoglu, K., Yang, J., Ulukus, S., and Yener, A. (2011). Transmission with energy harvesting nodes in fading wireless channels: Optimal policies. *IEEE Journal on Selected Areas in Communications*, 29(8):1732–1743.

[Paradiso and Starner, 2005] Paradiso, J. A. and Starner, T. (2005). Energy scavenging for mobile and wireless electronics. *IEEE Pervasive computing*, 4(1):18–27.

[Parisi et al., 2014] Parisi, S., Pirotta, M., Smacchia, N., Bascetta, L., and Restelli, M. (2014). Policy gradient approaches for multi-objective sequential decision making. In *2014 International Joint Conference on Neural Networks (IJCNN)*, pages 2323–2330. IEEE.

[Pathak et al., 2017] Pathak, D., Agrawal, P., Efros, A. A., and Darrell, T. (2017). Curiosity-driven exploration by self-supervised prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 16–17.

[Pathak et al., 2019] Pathak, D., Gandhi, D., and Gupta, A. (2019). Self-supervised exploration via disagreement. *arXiv preprint arXiv:1906.04161*.

[Peng and Low, 2014] Peng, S. and Low, C. (2014). Prediction free energy neutral power management for energy harvesting wireless sensor nodes. *Ad Hoc Networks*, 13:351–367.

[Pirotta et al., 2015] Pirotta, M., Parisi, S., and Restelli, M. (2015). Multi-objective reinforcement learning with continuous pareto frontier approximation. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*.

[Qiu et al., 2019] Qiu, C., Hu, Y., Chen, Y., and Zeng, B. (2019). Deep deterministic policy gradient (ddpg)-based energy harvesting wireless communications. *IEEE Internet of Things Journal*, 6(5):8577–8588.

[Raghunathan et al., 2005] Raghunathan, V., Kansal, A., Hsu, J., Friedman, J., and Srivastava, M. (2005). Design considerations for solar energy harvesting wireless embedded systems. In *Proceedings of the 4th international symposium on Information processing in sensor networks*, page 64. IEEE Press.

[Rahimi et al., 2003] Rahimi, M., Shah, H., Sukhatme, G. S., Heideman, J., and Estrin, D. (2003). Studying the feasibility of energy harvesting in a mobile sensor

network. In *Robotics and Automation, 2003. Proceedings. ICRA'03. IEEE International Conference on*, volume 1, pages 19–24. IEEE.

[Reddy and Murthy, 2012] Reddy, S. and Murthy, C. R. (2012). Dual-stage power management algorithms for energy harvesting sensors. *IEEE Transactions on Wireless Communications*, 11(4):1434–1445.

[Restuccia and Melodia, 2020] Restuccia, F. and Melodia, T. (2020). Deepwierl: Bringing deep reinforcement learning to the internet of self-adaptive things. pages 844–853.

[Rioual et al., 2018] Rioual, Y., Le Moullec, Y., Laurent, J., Khan, M. I., and Diguet, J.-P. (2018). Reward function evaluation in a reinforcement learning approach for energy management. In *2018 16th Biennial Baltic Electronics Conference (BEC)*, pages 1–4. IEEE.

[Rucco et al., 2013] Rucco, L. et al. (2013). A bird's eye view on reinforcement learning approaches for power management in wsns. In *6th WMNC*, pages 1–8.

[Rumelhart et al., 1986] Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning representations by back-propagating errors. *nature*, 323(6088):533–536.

[Salimans et al., 2017] Salimans, T., Ho, J., Chen, X., Sidor, S., and Sutskever, I. (2017). Evolution strategies as a scalable alternative to reinforcement learning. *arXiv preprint arXiv:1703.03864*.

[Schaul et al., 2015] Schaul, T., Quan, J., Antonoglou, I., and Silver, D. (2015). Prioritized experience replay. *arXiv preprint arXiv:1511.05952*.

[Schulman et al., 2015] Schulman, J., Levine, S., Abbeel, P., Jordan, M., and Moritz, P. (2015). Trust region policy optimization. In *International Conference on Machine Learning*, pages 1889–1897.

[Schurgers and Srivastava, 2001] Schurgers, C. and Srivastava, M. B. (2001). Energy efficient routing in wireless sensor networks. In *Military communications conference, 2001. MILCOM 2001. Communications for network-centric operations: Creating the information force. IEEE*, volume 1, pages 357–361. IEEE.

[Shah and Rabaey, 2002] Shah, R. C. and Rabaey, J. M. (2002). Energy aware routing for low energy ad hoc sensor networks. In *Wireless Communications and Networking Conference, 2002. WCNC2002. 2002 IEEE*, volume 1, pages 350–355. IEEE.

[Sharma et al., 2019] Sharma, M. K., Zappone, A., Assaad, M., Debbah, M., and Vassilaras, S. (2019). Distributed power control for large energy harvesting networks: A multi-agent deep reinforcement learning approach. *IEEE Transactions on Cognitive Communications and Networking*, 5(4):1140–1154.

[Sharma et al., 2010a] Sharma, N. et al. (2010a). Cloudy computing: Leveraging weather forecasts in energy harvesting sensor systems. In *7th IEEE SECON*, pages 1–9.

[Sharma et al., 2010b] Sharma, V., Mukherji, U., Joseph, V., and Gupta, S. (2010b). Optimal energy management policies for energy harvesting sensor nodes. *IEEE Transactions on Wireless Communications*, 9(4):1326–1336.

[Shelton, 2001] Shelton, C. R. (2001). Importance sampling for reinforcement learning with multiple objectives.

[Shih et al., 2001] Shih, E., Cho, S.-H., Ickes, N., Min, R., Sinha, A., Wang, A., and Chandrakasan, A. (2001). Physical layer driven protocol and algorithm design for energy-efficient wireless sensor networks. In *Proceedings of the 7th annual international conference on Mobile computing and networking*, pages 272–287. ACM.

[Shresthamali et al., 2017] Shresthamali, S., Kondo, M., and Nakamura, H. (2017). Adaptive power management in solar energy harvesting sensor node using reinforcement learning. *ACM Transactions on Embedded Computing Systems (TECS)*, 16(5s):181.

[Shresthamali et al., 2019] Shresthamali, S., Kondo, M., and Nakamura, H. (2019). Power management of wireless sensor nodes with coordinated distributed reinforcement learning. In *2019 IEEE 37th International Conference on Computer Design (ICCD)*, pages 638–647. IEEE.

[Silver et al., 2016] Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., et al. (2016). Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484.

[Silver et al., 2014] Silver, D., Lever, G., Heess, N., Degris, T., Wierstra, D., and Riedmiller, M. (2014). Deterministic policy gradient algorithms.

[Sindhya, 2011] Sindhya, K. (2011). Hybrid evolutionary multi-objective optimization with enhanced convergence and diversity. *Jyväskylä studies in computing*, (131).

[Singh et al., 1998] Singh, S., Woo, M., and Raghavendra, C. S. (1998). Power-aware routing in mobile ad hoc networks. In *Proceedings of the 4th annual ACM/IEEE international conference on Mobile computing and networking*, pages 181–190. ACM.

[Sinha and Chandrakasan, 2001] Sinha, A. and Chandrakasan, A. (2001). Dynamic power management in wireless sensor networks. *IEEE Design & Test of Computers*, 18(2):62–74.

[Sorokin et al., 2015] Sorokin, I., Seleznev, A., Pavlov, M., Fedorov, A., and Ignateva, A. (2015). Deep attention recurrent q-network. *arXiv preprint arXiv:1512.01693*.

[Sudevalayam and Kulkarni, 2011] Sudevalayam, S. and Kulkarni, P. (2011). Energy harvesting sensor nodes: Survey and implications. *IEEE Communications Surveys & Tutorials*, 13(3):443–461.

[Sundaresan et al., 2009] Sundaresan, S. et al. (2009). Event-driven adaptive duty-cycling in sensor networks. *International Journal of Sensor Networks*, 6(2):89–100.

[Sutton and Barto, 2018] Sutton, R. S. and Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT press.

[Sutton et al., 1992] Sutton, R. S. et al. (1992). Reinforcement learning is direct adaptive optimal control. *IEEE Control Systems*, 12(2):19–22.

[Thrun and Schwartz, 1993] Thrun, S. and Schwartz, A. (1993). Issues in using function approximation for reinforcement learning. In *Proceedings of the Fourth Connectionist Models Summer School*, pages 255–263. Hillsdale, NJ.

[Thrun, 1992] Thrun, S. B. (1992). E cient exploration in reinforcement learning. Technical report, Technical Report CMU-CS-92-102, School of Computer Science, Carnegie Mellon ....

[TMote, 2019] TMote (2019). Tmote sky. `http://www.eecs.harvard.edu/~konrad/projects/shimmer/references/tmote-sky-datasheet.pdf`. [Online; accessed 6-July-2019].

[Tokic and Palm, 2011] Tokic, M. and Palm, G. (2011). Value-difference based exploration: adaptive control between epsilon-greedy and softmax. In *Annual Conference on Artificial Intelligence*, pages 335–346. Springer.

[Ulukus et al., 2015] Ulukus, S., Yener, A., Erkip, E., Simeone, O., Zorzi, M., Grover, P., and Huang, K. (2015). Energy harvesting wireless communications: A review of recent advances. *IEEE Journal on Selected Areas in Communications*, 33(3):360–381.

[Vamplew et al., 2008] Vamplew, P., Yearwood, J., Dazeley, R., and Berry, A. (2008). On the limitations of scalarisation for multi-objective reinforcement learning of pareto fronts. In *Australasian Joint Conference on Artificial Intelligence*, pages 372–378. Springer.

[Van Hasselt et al., 2018] Van Hasselt, H., Doron, Y., Strub, F., Hessel, M., Sonnerat, N., and Modayil, J. (2018). Deep reinforcement learning and the deadly triad. *arXiv preprint arXiv:1812.02648*.

[Van Hasselt et al., 2016] Van Hasselt, H., Guez, A., and Silver, D. (2016). Deep reinforcement learning with double q-learning. In *Thirtieth AAAI Conference on Artificial Intelligence*.

[Van Huynh et al., 2019] Van Huynh, N., Hoang, D. T., Nguyen, D. N., Dutkiewicz, E., Niyato, D., and Wang, P. (2019). Optimal and low-complexity dynamic spectrum access for rf-powered ambient backscatter system with online reinforcement learning. *IEEE Transactions on Communications*, 67(8):5736–5752.

[Verma et al., 2006] Verma, A., Sawant, H., and Tan, J. (2006). Selection and navigation of mobile sensor nodes using a sensor network. *Pervasive and Mobile Computing*, 2(1):65–84.

[Vigorito et al., 2007] Vigorito, C. M., Ganesan, D., and Barto, A. G. (2007). Adaptive control of duty cycling in energy-harvesting wireless sensor networks. In *2007 4th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks*, pages 21–30. IEEE.

[Vinyals et al., 2019] Vinyals, O., Babuschkin, I., Chung, J., Mathieu, M., Jaderberg, M., Czarnecki, W. M., Dudzik, A., Huang, A., Georgiev, P., Powell, R., Ewalds, T., Horgan, D., Kroiss, M., Danihelka, I., Agapiou, J., Oh, J., Dalibard, V., Choi, D., Sifre, L., Sulsky, Y., Vezhnevets, S., Molloy, J., Cai, T., Budden, D., Paine, T., Gulcehre, C., Wang, Z., Pfaff, T., Pohlen, T., Wu, Y., Yogatama, D., Cohen, J., McKinney, K., Smith, O., Schaul, T., Lillicrap, T., Apps, C., Kavukcuoglu, K., Hassabis, D., and Silver, D. (2019). Alphastar: Mastering the real-time strategy game starcraft ii. `https://deepmind.com/blog/alphastar-mastering-real-time-strategy-game-starcraft-ii/`.

[Wang and Wang, 2006] Wang, P. and Wang, T. (2006). Adaptive routing for sensor networks using reinforcement learning. In *The Sixth IEEE International Conference on Computer and Information Technology (CIT'06)*, pages 219–219. IEEE.

[Wang and Sebag, 2012] Wang, W. and Sebag, M. (2012). Multi-objective Monte-Carlo tree search. In Hoi, S. C. H. and Buntine, W., editors, *Proceedings of the Asian Conference on Machine Learning*, volume 25 of *Proceedings of Machine Learning Research*, pages 507–522, Singapore Management University, Singapore. PMLR.

[Wang et al., 2015a] Wang, X., Gong, J., Hu, C., Zhou, S., and Niu, Z. (2015a). Optimal power allocation on discrete energy harvesting model. *EURASIP Journal on Wireless Communications and Networking*, 2015(1):48.

[Wang et al., 2011] Wang, Y., Xie, Q., Ammari, A., and Pedram, M. (2011). Deriving a near-optimal power management policy using model-free reinforcement learning and bayesian classification. In *Proceedings of the 48th Design Automation Conference*, pages 41–46. ACM.

[Wang et al., 2015b] Wang, Z., Schaul, T., Hessel, M., Van Hasselt, H., Lanctot, M., and De Freitas, N. (2015b). Dueling network architectures for deep reinforcement learning. *arXiv preprint arXiv:1511.06581*.

[Warrier and Kumar, 2016] Warrier, M. M. and Kumar, A. (2016). Energy efficient routing in wireless sensor networks: A survey. In *Wireless Communications, Signal Processing and Networking (WiSPNET), International Conference on*, pages 1987–1992. IEEE.

[Watkins and Dayan, 1992] Watkins, C. J. and Dayan, P. (1992). Q-learning. *Machine learning*, 8(3-4):279–292.

[Watkins, 1989] Watkins, C. J. C. H. (1989). *Learning from delayed rewards*. PhD thesis, King's College, Cambridge.

[Webb and Hatton, 2020] Webb, W. and Hatton, M. (2020). *The Internet of Things Myth*.

[Wei et al., 2017] Wei, T., Wang, Y., and Zhu, Q. (2017). Deep reinforcement learning for building hvac control. In *Design Automation Conference (DAC), 2017 54th ACM/EDAC/IEEE*, pages 1–6. IEEE.

[Werner-Allen et al., 2006] Werner-Allen, G., Lorincz, K., Ruiz, M., Marcillo, O., Johnson, J., Lees, J., and Welsh, M. (2006). Deploying a wireless sensor network on an active volcano. *IEEE internet computing*, 10(2):18–25.

[Wu et al., 2017] Wu, F., Rüdiger, C., and Yuce, M. R. (2017). Real-time performance of a self-powered environmental iot sensor network system. *Sensors*, 17(2):282.

[Xiao et al., 2015] Xiao, Y., Han, Z., Niyato, D., and Yuen, C. (2015). Bayesian reinforcement learning for energy harvesting communication systems with uncertainty. In *Communications (ICC), 2015 IEEE International Conference on*, pages 5398–5403. IEEE.

[Xu et al., 2018] Xu, Y., Lee, H. G., Chen, X., Peng, B., Liu, D., and Liang, L. (2018). Puppet: Energy efficient task mapping for storage-less and converter-less solar-powered non-volatile sensor nodes. In *2018 IEEE 36th International Conference on Computer Design (ICCD)*, pages 226–233. IEEE.

[Xu et al., 2019] Xu, Y., Lee, H. G., Tan, Y., Wu, Y., Chen, X., Liang, L., Qiao, L., and Liu, D. (2019). Tumbler: Energy efficient task scheduling for dual-channel solar-powered sensor nodes. In *Proceedings of the 56th Annual Design Automation Conference 2019*, DAC '19, pages 172:1–172:6, New York, NY, USA. ACM.

[Yang et al., 2019] Yang, R., Sun, X., and Narasimhan, K. (2019). A generalized algorithm for multi-objective reinforcement learning and policy adaptation. In *Advances in Neural Information Processing Systems*, pages 14636–14647.

[Yau et al., 2012] Yau, K.-L. A., Komisarczuk, P., and Teal, P. D. (2012). Reinforcement learning for context awareness and intelligence in wireless networks: Review, new features and open issues. *Journal of Network and Computer Applications*, 35(1):253–267.

[Yick et al., 2008] Yick, J., Mukherjee, B., and Ghosal, D. (2008). Wireless sensor network survey. *Computer networks*, 52(12):2292–2330.

[Younis et al., 2002] Younis, M., Youssef, M., and Arisha, K. (2002). Energy-aware routing in cluster-based sensor networks. In *Modeling, Analysis and Simulation of Computer and Telecommunications Systems, 2002. MASCOTS 2002. Proceedings. 10th IEEE International Symposium on*, pages 129–136. IEEE.

[Zeng et al., 2010] Zeng, F., Zong, Q., Sun, Z., and Dou, L. (2010). Self-adaptive multi-objective optimization method design based on agent reinforcement learning for elevator group control systems. In *2010 8th World Congress on Intelligent Control and Automation*, pages 2577–2582. IEEE.

[Zhang et al., 2004] Zhang, P., Sadler, C. M., Lyon, S. A., and Martonosi, M. (2004). Hardware design experiences in zebranet. In *Proceedings of the 2nd international conference on Embedded networked sensor systems*, pages 227–238.

[Zhao and Gurusamy, 2008] Zhao, Q. and Gurusamy, M. (2008). Lifetime maximization for connected target coverage in wireless sensor networks. *IEEE/ACM transactions on networking*, 16(6):1378–1391.