**UNIVERSITY OF SOUTHAMPTON**
Faculty of Engineering, Science and Mathematics
School of Electronics and Computer Science

# Multi–Armed Bandit Models for Efficient Long–Term Information Collection in Wireless Sensor Networks

by Long Tran–Thanh

Supervisors: Nicholas R. Jennings and Alex Rogers
Examiner: Jörg Fliege

A Mini–Thesis Submitted for Transfer from MPhil to PhD

April 14, 2010

UNIVERSITY OF SOUTHAMPTON

ABSTRACT

FACULTY OF ENGINEERING, SCIENCE AND MATHEMATICS
SCHOOL OF ELECTRONICS AND COMPUTER SCIENCE

A mini–thesis submitted for transfer from MPhil to PhD

by Long Tran–Thanh

We are entering a new age in the evolution of computer systems, in which pervasive computing technologies seamlessly interact with human users. These technologies serve people in their everyday lives at home and work by functioning invisibly in the background, creating a smart environment around them. For example, this could be an intelligent building or a smart traffic control system. Now, since such smart environments need information about their surroundings to function effectively, they rely first and foremost on sensory data from the real world. More accurately, this data is typically provided by wireless sensor networks, which are networks of small, autonomous sensor devices.

The advantages of wireless sensor networks, such as flexibility, low cost and ease of deployment, have ensured they have gained significant attention from both researchers and manufacturers. However, due to the limited resource constraints of such sensors (e.g. hardware limitations, low computational capacity, or limited energy budget), there are still a number of significant and specific research challenges to be addressed in this domain.

To overcome these challenges, we believe an efficient solution for long–term information collection in wireless sensor network should be able to fulfill the following requirements: (i) adaptivity to environmental changes; (ii) robustness and flexibility; (iii) computational feasibility; and (iv) limited use of communication. In more detail, wireless sensor networks are typically deployed in dynamic environments, we must take environmental changes into account, and thus, it must be able to *adapt* to those changes. Furthermore, since future changes of the environment are typically unknown *a priori*, we cannot accurately predict these changes. Thus, in order to efficiently adapt to the environment, a good solution must be *on–line*, so that it can quickly react to environmental changes. Besides, we must be aware of topological and physical changes (e.g. node or communication failures) as well. Finally, due to the limited resources of the sensors, communication and computational cost should not be significant, compared to the size of the network.

Previous work of information collection in wireless sensor networks has typically focused on optimising data sampling, routing, information valuation and energy management

in order to achieve efficient information collection. However, it usually fail to provide all of the aforementioned requirements. Specifically, existing solutions are typically not designed for long–term operation, since they cannot adapt to environmental changes. That is, they do not have the ability of modifying their behaviour so that they could efficiently adapt to the new characteristics of the environment. Other algorithms follow the concept of centralised control mechanism (i.e. a central unit is responsible for all the calculations and decision making). These solutions, however, are not robust and flexible, since the central unit may represent a computational bottleneck.

Against this background, this transfer report focuses on the challenge of developing decentralised adaptive on–line algorithms for efficient long–term information collection in the wireless sensor network domain. In particular, we focus on developing energy management and information–centric data routing policies that adapt their behaviour according to the energy that is harvested, in order to achieve efficient performance. In so doing, we introduce two new energy management techniques, based on multi–armed bandit learning, that allow each sensor to adaptively allocate its energy budget across the tasks of data sampling, receiving and transmitting. These approaches are devised in order to deal with the following different situations: (i) when the sensors can harvest energy from the environment; and (ii) when energy harvesting from the environment is not possible. By using this approaches, each sensor can learn the optimal energy budget settings that gives it efficient information collection in the long run. In addition, we propose a novel decentralised algorithm for information–centric routing.

In more detail, we first tackle the energy management problem with energy–harvesting sensors from the multi–armed bandit perspective. That is, we reduce the energy management problem to a non–stochastic multi–armed bandit model. Then through extensive simulations, we demonstrate that the performance of this approach outperforms other state–of–the–art non–learning algorithms. For the case of energy management with non–harvesting sensors, we show that existing multi–armed bandit models are not suitable for modelling this problem. Given this, we introduce a new bandit model, the budgeted multi–armed bandit with pulling cost, in order to efficiently tackle the energy management problem. Following this, we propose an $\epsilon$–first approach for this new bandit problem, in which the first $\epsilon$ portion of the total budget is allocated to exploration (i.e. learning which actions are the most efficient). Finally, for the routing, we introduce an information–centric routing problem, the maximal information throughput routing problem. Existing routing algorithms, however, are not suitable to solve this problem. Thus, we devise a simple, but proveably optimal decentralised algorithm, that maximises the information throughput in the network.

# Contents

# List of Figures

# List of Algorithms

# List of Acronyms

**BS** Base station

**Exp3** Exponential-weight algorithm for exploration and exploitation

**MAB** Multi-armed bandit

**MITRA** Maximal information routing algorithm

**RL** Reinforcement learning

**UCB** Upper confidence bound

**WSN** Wireless sensor network

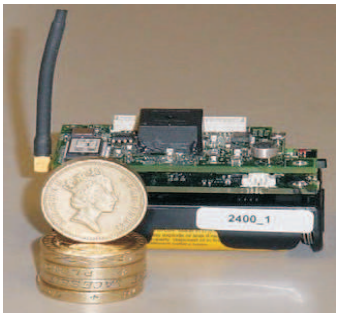
# Chapter 1

# Introduction

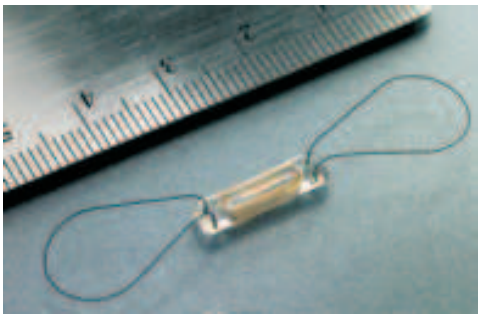
Due to their flexibility, low cost and ease of deployment, networks of wireless sensors are being widely used in a wide variety of applications ranging from environmental, habitat and traffic monitoring, to object tracking and military field observations [Chong and Kumar, 2003; Merrett, 2008; Rogers *et al.*, 2009; Romer and Mattern, 2004]. Specifically, in this context, a *wireless sensor network* (WSN) is viewed as a network of small, densely deployed, spatially decentralised autonomous sensor devices (referred to hereafter as "nodes") communicating through a wireless communication network, whose task is monitoring physical or environmental conditions (including, but not limited to, the following: temperature, sound, vibration, pressure, seismic, infrared, magnetic and motion information) [Baldus *et al.*, 2004; Juang *et al.*, 2002; Simon *et al.*, 2004].

In WSNs, each wireless sensor node is typically equipped with a sensing module for sensing data from the surrounding environment, a radio transceiver module for wireless communication, a small microcontroller as the processing unit, an external memory for data storage and a limited energy source (usually a battery). The size of a single sensor node can vary from the size of a shoe box to the size of a coin (see Figure 1.1). Furthermore, their cost is similarly variable, ranging from hundreds of pounds to a few pence, constrained by parameters such as size, energy, memory, computational speed and communication bandwidth required of individual nodes [Romer and Mattern, 2004].

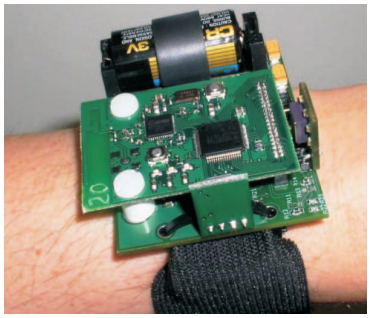
These networks are typically deployed for *collecting information* from a heterogeneous (i.e. the environmental characteristics may vary over space) and dynamically changing (i.e. the characteristics may vary over time) environment, and are typically required to operate over an *extended period of time* (covering months or even years). The collected information is typically forwarded to a *base station* ($BS$) (also referred to as a sink or gateway), using single-hop (i.e. sensors directly send data to the $BS$) or multi-hop (i.e. data is forwarded towards the $BS$ via relay nodes) routing paths. In most cases, the $BS$ is a specialised node that has significantly more power than the ordinary ones and,




a) wireless sensor mote[1]



b) wireless cardio sensor[2]



c) wearable sensor[3]

d) robot football[4]

FIGURE 1.1: Wireless sensor nodes.

depending on the WSN application, there can be more than one of them in the WSN (see Figure 1.2). Information that arrives at the $BS$ is then processed or transmitted outside the sensor network to end-users for further analysis, fulfilling the goals of the WSN deployment. Given this, in this report, we focus in particular on the challenges associated with *efficient long-term information collection* in WSNs. That is, *we aim to maximise the total information value collected and delivered to the $BS$ in a given extended time interval of operation, or over the entire lifetime of the network.*

Now, to efficiently control the information collection in the network, a protocol is needed to dictate the actions of the sensor nodes [Kho, 2009]. Such protocols fall into two broad categories: namely *centralised* and *decentralised*. In the former, a single controller receives information from all the nodes, and then determines the actions of each node indicating how they should sample, receive, forward, and route data. Within this approach, however, the central controller is required to perform a large number of computations in order to find each node's optimal actions. Thus, it could represent a significant computational bottleneck, especially in the case of large network size. Furthermore, since the controller needs to collect all the information from the nodes, there typically is a delay

[1]Taken from link `http://www.npl.co.uk/server.php?show=ConWebDoc.289`;
[2]Taken from link `http://amp.osu.edu/news/article.cfm?ID=4576`;
[3]Taken from link `http://www.intelligent-systems.info/biofeedback/biofeedback.htm`;
[4]Taken from link `http://amp.osu.edu/news/article.cfm?ID=4576`.
All links are checked on 14/04/2010.

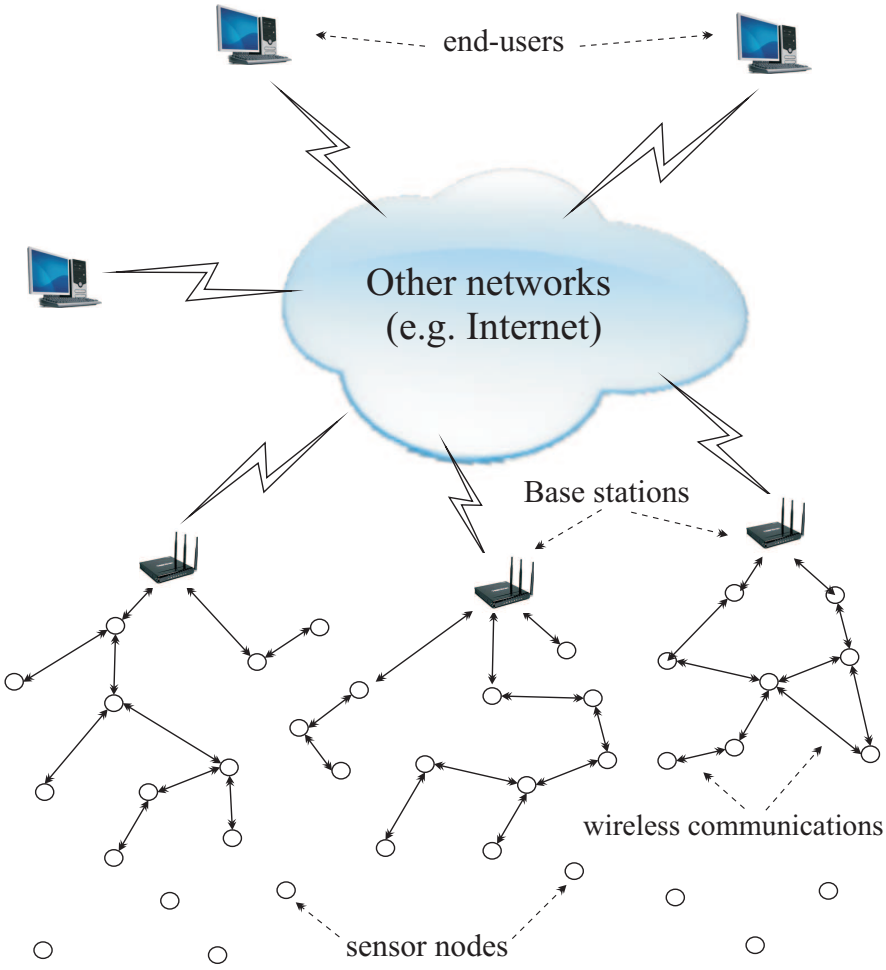



FIGURE 1.2: Typical wireless sensor network.

in time before it can start determining the nodes' actions. Given this, the centralised approach cannot respond well to the environmental changes, and thus, it suffers from a lack of ability to efficiently adapt to those changes. In contrast, in decentralised control regimes such a central controller does not exist. Instead, the nodes are autonomous and each decides its individual actions based on its own local state and observations, and those of its neighbouring nodes. Thus, such aforementioned disadvantages disappear within the decentralised control regime [Boukerche, 2008; Wagner and Wattenhofer, 2007]. Therefore, in this report, we focus on decentralised control mechanism for WSNs.

The decentralised approach, however, has several additional challenges. Specifically, to achieve system-wide goals, the nodes must typically coordinate their actions with their neighbours (e.g. to forward data or to track objects). In addition, since the nodes typically operate in a dynamically changing environment, they must be able to *autonomously* adapt their behaviour to environmental changes, in order to achieve long-term global goals (e.g. maximal data collection or optimal coverage). Such issues naturally lend themselves to a *multi-agent system* (MAS) perspective [Lesser *et al.*, 2003; Rogers *et al.*,

2009], in which each sensor is represented by an agent, which autonomously and cooperatively acts, in order to achieve system-wide objectives [Jennings, 2001].

However, the exact mapping between a WSN and a MAS is often non–trivial. In particular, MAS do not typically address physical resource constraints such as the computational, communication and energy limits of the agents (i.e. nodes); whereas WSNs are heavily resource constrained (i.e. low energy capacity, size and computational constraints) [Akyildiz *et al.*, 2002; Rogers *et al.*, 2009]. Thus, there are a number of significant and specific research challenges to be addressed. These include, but are not limited to, the following: hardware challenges (e.g. cost and size of sensor nodes), time synchronisation, lossy wireless communication, node failures, energy efficiency, and information processing [Chong and Kumar, 2003; Stankovic, 2004].

To overcome these shortcomings, researchers have proposed a number of solutions from different perpectives, in order to achieve efficient information collection in WSNs.

From the perspective of sampling, *adaptive data sampling* policies have been advocated as the way to achieve accurate estimates of the environmental conditions, while minimising redundant sampling of the environment [Jain and Chang, 2004; Kho *et al.*, 2009a; Padhy *et al.*, 2006; Willett *et al.*, 2004]. In particular, adaptive sampling policies often include sets of rules that control a node's *sampling rate* (i.e. how often a node is required to collect data by sampling during a particular time interval) and *sampling scheduling* (i.e. when a node is required to sample). The advantage of adaptive sampling is that it can efficiently deal with the environmental changes by adaptively changing the sampling rate, and thus, is capable of achieving good performance in the long–term. Given this, we aim to use an efficient adaptive method for data sampling in this report.

Within the routing context, researchers have developed *information-centric routing* protocols for forwarding data. These routing techniques are usually aimed at maintaining a high *throughput* (i.e. rate of successful message delivery) of information towards the $BS$ during the operation of the network. In particular, a data forwarding policy determines actions that an agent can use to forward data towards the $BS$, namely (i) *transmitting* (i.e. which *data packets* the agent should choose to transmit, and to which node), and (ii) *receiving* (i.e. how many packets an agent is required to receive from the other agents during a transmission period) tasks. Here, data packets can be considered as units of data with the same size. In addition, since it is generally accepted that the energy consumption of the communication module is the most significant drain on the sensor node's energy resources, communicating trivial or redundant data across a network constitutes a waste of resources [Anastasi *et al.*, 2004; Mathur *et al.*, 2006]. Thus, to achieve a better performance in data collection, it is necessary to consider the usefulness of data to the end-user by defining different levels of *importance of information* (i.e. information value), as opposed to treating all data as having homogeneous usefulness

[Deb *et al.*, 2003; Merrett, 2008]. In so doing, *information content valuation* techniques are typically used to calculate the importance of data [Frieden, 2004; Kho, 2009; Krause *et al.*, 2006]. In particular, an information value metric, such as Fisher information or mutual information is used to determine the level of importance of information. Given this, information-centric routing methods give more support to the delivery of important data (i.e. data with higher information value), and are only concerned with the less important data (i.e. data with lower information value) when resources allow. Therefore, they can maximise the total information value delivered to the $BS$. Thus, in this report, we also pursue an information-centric approach for routing.

From the perspective of energy management, such solutions are commonly said to be *energy-aware* [Chandrakasan *et al.*, 2002; Raghunathan *et al.*, 2002; Stankovic, 2004]. That is, it is necessary to efficiently manage the energy consumption of the agents. Otherwise, rapid battery depletion may lead to insufficient data collection from the network. In particular, a WSN with a longer lifetime typically increases the amount of information collected by the network. Recently, however, a number of research works have proposed *energy harvesting sensors*, which have the capability of scavenging ambient energy from their surrounding environment, using solar, vibration, temperature, and radioactive sources [Beeby *et al.*, 2006; Merrett, 2008; Roundy *et al.*, 2004; Torah *et al.*, 2008; Zhang *et al.*, 2004]. Within such applications, sensor agents typically seek to comply with the concept of *energy–neutrality*, in which the energy consumption of a node should be equal to the harvested energy [Kansal and Srivastava, 2003; Kho *et al.*, 2009a; Vigorito *et al.*, 2007]. The advantage of such approaches is that agents can indefinitely extend their life span, which is especially important when information collection has to operate over a prolonged period of time. Since the focus of this report is on the efficiency of long–term performance, we use sensors with rechargeable batteries. Furthermore, in our work, these sensor agents follow the concept of energy–neutrality.

In addition, since WSNs have to operate over an extended period of time, within dynamic environments, it is necessary for the agents to *adapt* to environmental changes, in order to achieve efficient long-term information collection [Predd *et al.*, 2006]. That is, by adaptively changing their focus on each action type (i.e. tasks), the agents can decide whether to put more effort on sampling (e.g. when significant events are occurring in the monitored area), receiving important data from the others (e.g. when they have collected high value information that has to be delivered to the $BS$), or transmitting data (e.g. when the delivery of data cannot be delayed too long). However, in order to determine the optimal action type (*exploitation*), the agent first has to *learn* the performance of other action types as well (*exploration*). That is, the agent has to efficiently deal with the trade–off between exploration and exploitation. Furthermore, the agent must learn on–line, since the environment is typically unknown a priori for the network. In the literature, efficient and robust approaches have been designed to efficiently deal with

these challenges, such as *multi–armed bandit* learning [Robbins, 1952], or *reinforcement learning* [Sutton and Barto, 1998]. In the former type of learning, each agent chooses from its action set at each round, and based on this choice, it gets a reward. The goal of the agent is simply to maximise its total reward over a given time period. The latter type of learning can be regarded as the more generic form of the former, since the agent can also modify the environment's state, by taking its actions (for more details see sections 2.2 and 2.3). By using these approaches, agents should be able to adapt to environmental changes, in order to perform efficient long-term information collection. Given this, in order to efficiently handle the environmental dynamism, we pursue a learning approach for energy management.

To date, however, none of the previous work brings all these diverse aspects of the problem together into an integrated framework that is necessary to make progress in this area (see chapter 2 for more details). Thus, this work seeks to start addressing this gap. Specifically, in section 1.1, we describe the research requirements that information collection mechanisms should satisfy, in order to achieve efficient long-term performance. Following that, we introduce the contributions of this report in section 1.2. Finally, section 1.3 outlines the overall structure of the remainder of this report.

## 1.1 Research Requirements

The aim of the work in this report is to design efficient mechanisms to maximise long-term information collection in WSNs. Such mechanisms, however, have to deal with a number of issues, such as a constantly changing environment and significant resource constraints. Given this, we can define the following broad requirements that information collection mechanisms for WSNs should satisfy:

1. **Adaptivity**: Since the nodes are typically deployed in dynamic environments, efficient performance cannot be sustained without the ability to learn and to adapt to environmental changes. That is, the nodes should be able to learn efficient policies on–line, based on their own experiences. Moreover, they have to achieve an efficient trade-off between exploration and exploitation.

2. **Robustness and flexibility**: Due to the long operating time of the network, node failures and lossy communication links are likely to occur. Even in these cases, the network operation should not collapse, but degrades gracefully. Therefore, the nodes should be able to handle these situations, by ensuring that the operation of the remaining nodes is not affected.

3. **Computational feasibility**: Due to the hardware constraints (e.g. limited computational capacity or memory size), information collection approaches with significant computational cost are not affordable here. Consequently, it is necessary to develop efficient algorithms that consume small amounts of computational resources.

4. **Limited use of communication**: Since communication is typically the most expensive task in WSNs, a good information collection approach should avoid having significant communication cost (i.e. the energy amount allocated to sending control messages), in order to achieve efficient performance in information collection. In particular, by reducing the amount of energy for communication, the sensors can allocate more energy to forwarding real data, and thus, it can increase the amount of collected information.

A good information collection approach for WSNs will comply with all four of these requirements. Therefore, the design of such approaches should satisfy all of these requirements. In this report we show that the approach of using multi–armed bandit learning based energy management and maximal information throughput routing satisfy these concerns. In so doing, we compare the performance of our approach with that of other techniques from the broader literature on reinforcement learning and information–centric routing. The specific contributions of this report are now discussed in more detail.

## 1.2 Research Contributions

The main focus of this report is designing information collection approaches for WSNs, that provide efficient performance in a prolonged period of time, satisfying the research requirements of: (i) adaptivity; (ii) robustness and flexibility; (iii) computational feasibility; and (iv) limited use of communication.

Specifically, the information collection problem that we address consists of a set of sensor agents with rechargeable batteries, collecting information from a dynamic environment over an extended period of time, without the aid of a centralised controller. The collected information is then delivered to the $BS$, using information-centric routing techniques and energy–neutral energy management. Here, we focus on energy management and information-centric routing, while we assume that efficient sampling and information content valuation can be achieved by using existing techniques. Thus, to simplify the complexity of the original problem, we separately develop energy management and routing policies. However, as we will show, efficient data collection can be still achieved by combining such policies together.

In particular, for the *energy management problem* (i.e. how to adaptively allocate energy budgets to each of the sensory tasks), we pursue a simple reinforcement learning approach, namely *multi-armed bandit* (MAB) learning [Robbins, 1952]. That is, each agent's energy budget allocation problem is represented as a MAB problem. There exist other, more sophisticated, learning methods, such as multi-state Markov decision processes (MDPs) [Sutton and Barto, 1998], partially observable MDPs (POMDPs) [Cassandra, 1998], or decentralised POMDPs (DecPOMDPs) [Seuken and Zilberstein, 2008], which can also be used here. However, as we will see later, the MAB approach produces remarkably good results in information collection. Given this, based on MAB learning, we devise two energy management algorithms, each of them dealing with a different type of energy budget limitation. In particular, from the aspect of energy recharging, the following situations may occur: (i) the agent's energy harvesting performance is stable (e.g. agent with solar battery is deployed outdoors, where the solar panel is not blocked from the sun light); and (ii) it may happen that the agent loses its ability to recharge its battery (e.g. the solar battery unit is malfunctioning, which may occur during a long period of operation), and thus, the agent's battery will not be recharged at all. In particular, in the first case, the energy budget of the agent is recharged each round, and thus, it can comply the concept of energy neutrality. On the other hand, in the second case, the agent's overall energy budget (i.e. the total energy budget for the whole operation time) is limited, and thus, energy management has to be energy-aware. For the first case, existing MAB models can be used to tackle the energy management problem (for more details, see chapters 5). However, the second case cannot be approached by those existing MAB models. Given this, we propose a new MAB model: the budgeted multi-armed bandit with pulling cost. By using this model, the case of limited overall budget can also be dealt with efficiently (for more details, see chapters 6).

Now, suppose that all the agents have already set their energy budget value for sampling, receiving, and transmitting tasks. In this case, to maximise the value of the total collected information, it is obvious that we need to maximise the total information value of data sampled or relayed by agents that are one hop from the $BS$. The latter, however, is equal to data that is sampled or relayed by agents that are two hops from the $BS$, and so on. That is, beside focusing on efficient energy management, it is also important to maximise the information throughput (i.e. the total transmitted information value) between neighbouring layers of agents (i.e. group of agents with the same distance from the $BS$) by using efficient routing techniques. Hereafter, for our convenience, we refer to this routing problem as the *maximal information throughput routing problem*. In order to solve this problem, we propose a simple, but proveably optimal, decentralised routing method, the maximal information throughput routing algorithm, for information–centric routing.

Thus, given this context, the work in this report advances the-state-of-the-art in the following specific ways:

**Maximal Information Throughput Routing (chapter 4)** – We develop a computationally simple decentralised information–centric routing algorithm, called MITRA (for maximal information throughput routing algorithm), for forwarding data packets towards the $BS$ (contribution 1). This is the first algorithm that proveably achieves optimal solution for the maximal information throughput routing problem. We also demonstrate that MITRA has low communication cost compared to the size of the network, by running extensive simulations.

**Energy Management With Energy Harvesting Sensors (chapter 5)** – We propose a novel, multi–armed bandit based, energy management algorithm for the case of agents with energy harvesting ability (contribution 2). This learning-based energy management policy operates on each agent, managing the agent's energy budget adaptively to achieve efficient long-term information collection. This algorithm is responsible for allocating energy budgets to the agent's tasks such as sampling, transmitting and receiving. Here, we use an existing MAB model, the non–stochastic MAB, which has been shown to be efficient in dynamic environments. Since the energy management task should be adaptive to the dynamics of the system, by using a learning approach, each agent is able to react to changes in the network and environment, learning to behave efficiently in different circumstances. To date, none of the state-of-the-art studies has used a learning-approach for long–term information collection. Thus, this algorithm is the first attempt to exploit the advantages of learning theory for efficient information collection in WSNs. In addition, we show that this algorithm has low computational and communication cost compared to the size of the network.

**Energy Management With Non-Harvesting Sensors (chapter 6)** – Since energy management in the case of non–harvesting sensors uses a concept that is different from energy–neutrality (which is used in the case of energy–harvesting sensors), existing MAB models do not suit this problem. Given this, we introduce a new MAB problem, the budgeted multi–armed bandit with pulling cost (contribution 3). In this model, the agent chooses a costly action, and the total budget for choosing actions (i.e. pulling arms) is limited. To tackle this problem, we propose budgeted $\epsilon$-first action choosing policies, for the case of static environments. Then we devise the first, theoretically proven upper bound for the loss of a budgeted $\epsilon$–first algorithm that uses any generic exploration method for this problem (contribution 4). Following this, we improve this upper bound for the case of budgeted $\epsilon$–first approach with uniform pull exploration, in which all of the arms are uniformly pulled in the exploration phase. Then we compare the efficiency of

budgeted $\epsilon$–first policies to other existing reinforcement learning techniques. With this comparison, we show that, within the same settings (i.e. pulling arms is costly, and the overall budget is limited), the proposed MAB model is more efficient in learning than other existing models. Following this, we propose a MAB approach for energy management with non–harvesting sensors. This approach is based on the proposed MAB model.

As can be seen, both the proposed MAB learning energy management methods and MITRA have low computational and communication cost, and thus, they satisfy requirements 3 and 4. Furthermore, by using the MAB learning approach, energy management can be adaptive to the environmental changes, and thus, it satisfy the adaptivity requirement. Besides, as later we will show, MITRA is capable of fulfilling the robustness and flexibility requirement. Given this, the combined use of the proposed methods satisfies the research requirements. In addition, through extensive simulations, we show that performance of long–term information collection can be significantly improved by using the proposed MAB learning energy management with the MITRA together, compared to that of other non–learning approaches.

The work up till now has led to the publication of two papers, which form the basis of chapters 4, 5, and 6, respectively:

- L. Tran–Thanh, A. Rogers, and N. R. Jennings. Adaptive Efficient Long-Term Information Collection with Energy Harvesting Wireless Sensors. *Submitted to*: The 19th European Conference on Artificial Intelligence (ECAI), Lisbon, Portugal, 2010.
- L. Tran-Thanh, A. Chapman, E. Munoz de Cote, A. Rogers, and N. R. Jennings. $\epsilon$–First Policies for Budget Limited Multi-Armed Bandits. In Proceedings of the 24th AAAI Conference on Artificial Intelligence (AAAI), Atlanta, USA, 2010.

## 1.3 Report Outline

The remainder of this report is structured as follows:

- Chapter 2 analyses the state-of-the-art in WSNs with respect to long–term information collection. Furthermore, we review the reinforcement learning and multi–armed bandit literature, respectively.
- Chapter 3 introduces our formal model of a WSN system. Following this, we formulate our research objectives, with respect to the research requirements of this

report. Then we describe the formalised definitions of the energy management and the maximal information throughput routing problems.

- Chapter 4 gives a decentralised algorithm for the maximal information throughput routing problem. First we give a formal problem description. Following that, we introduce our decentralised algorithm (MITRA). It is followed by the proof of its optimal performance in maximising the total information value of transmitted packets between neighbouring layers. Finally, we show simulation results that demonstrate the low communication cost of MITRA.

- Chapter 5 first introduces the problem of energy management with energy harvesting sensors. Then we describe the theoretical background of the non–stochastic multi–armed bandit problems. Next we discuss the standard Exp3 algorithm in more detail. Following that we formulate the energy management problem as a non–stochastic MAB problem, by defining the transition and reward functions, and we propose a MAB based energy management algorithm. We also analyse the performance of the proposed algorithm.

- Chapter 6 first describes the problem of energy management with non-harvesting sensor agents. Following that we introduce the model of budgeted multi–armed bandits with pulling cost, which is the basis of our energy management approach. Next we propose $\epsilon$–first policies for the static MAB model. Then we formulate the MAB based energy management problem, using the budgeted MAB with pulling cost model. Finally, we analyse the performance of our approach in detail.

- Chapter 7 presents our conclusions and outline the problems we would like to deal with as future work.

# Chapter 2

# Literature Review

In this chapter, we provide an overview of existing research studies against which our work is positioned. In order to do this, in the first part of the chapter (section 2.1) we discuss previous work on data collection of WSNs from different aspects, namely: adaptive sampling, information content valuation, information–centric routing, and energy management. Within these areas, we highlight the limitations of each of the proposed methods, motivating the goals of research in this report. Furthermore, as mentioned in chapter 1, WSNs must also be capable of learning optimal behaviours in unknown domains and capable of adapting to environmental changes in order to achieve efficient long-term performance in information collection. Thus, in the second part of the chapter (sections 2.2 – 2.4), we focus on the introduction of learning and long-term planning policies in WSNs. In addition, we look at related work in the WSN domain that also takes learning ability into account.

In more details, in section 2.1.1, we first detail some of the most commonly used adaptive sampling methods that have been developed for WSNs. In section 2.1.2, we provide a background review on information content valuation techniques. Following this, we discuss existing adaptive routing algorithms in section 2.1.3. Then, we focus on efficient energy management schemes for WSNs in section 2.1.4. We continue the literature review by introducing the basics of reinforcement learning (RL) theory in section 2.2. Following this, in section 2.3, we discuss the multi-armed bandit (MAB) problem, which can be regarded as a simplified RL. Then we describe a variety of multi–armed bandit models, namely: (i) non–stochastic MAB (section 2.3.1); and (ii) MAB with cost constraints (section 2.3.2). Then, the applications of RL techniques in the WSN domain are discussed in section 2.4. Finally, in section 2.5, we conclude this chapter by summarising our findings and relating them back to our original research requirements (as detailed in section 1.1).

## 2.1 Long–Term Information Collection in Wireless Sensor Networks

Here, we focus on describing existing approaches for information collection in WSNs. In particular, in the following sub–sections, we provide an overview of adaptive data sampling, information content valuation, information–centric routing, and energy management techniques, respectively.

### 2.1.1 Adaptive Data Sampling

In this section, we focus on adaptive sampling algorithms within the WSN domain. In dynamic environments, it is not sufficient to have sensors deployed with a fixed sampling rate, since such sensors may not be able to capture the rapid changes of their surroundings, and thus, their performance in information collection may not be efficient. Therefore, it is necessary to use some form of adaptive sampling approach on each sensor in the network. Generally speaking, adaptive sampling can be described as an "intelligent sampling", which is adaptive to the (dynamic) environmental changes in space and time. In particular, an adaptive sampling algorithm is here defined as a protocol (i.e. a set of policies) that is responsible for adaptively setting the sampling rate (i.e. *how often* a node is required to sample during a particular time interval) and schedule (i.e. *when* a node is required to sample) of each of the individual nodes in a network.

Existing algorithms can be classified as to whether they use temporal or spatial correlations (or both) in order to make effective sampling decisions. With respect to spatial correlations between sensors, the challenge of calculating informative locations has been thoroughly studied by Guestrin *et al.* [2005]. In this approach, the spatial correlations within the monitored environment are assumed to be known. That is, these correlations are modelled by using a Gaussian process and can be learnt during the initial deployment of the network. Based on this information, an informative subset of the sensors is selected to provide the information to a BS, while the rest of the nodes are removed in order to reduce cost. In addition, Krause *et al.* [2006] extend this work by taking communication cost into account, making an explicit trade-off between the energy consumption of sampling and the communicating of each sensor. Finally, Willett *et al.* [2004] have studied the backcasting adaptive sampling method in which multiple nodes that are spatially correlated form small subsets of nodes that then communicate their information to a local data aggregation coordinator. Based upon this information, the coordinator then selectively activates additional nodes (by instructing them to take samples) in order to achieve a target level of uncertainty. The first two techniques,

however, fail to give proper attention to the temporal dynamics of the monitored environment, and thus, they are essentially static (or single shot) approaches that do not explicitly consider long-term data collection. On the other hand, the third method uses a centralised coordination mechanism, that contains all the drawbacks of the centralised regime.

To handle temporal correlations, the *utility based sensing and communication* (USAC) algorithm was proposed by Padhy *et al.* [2006]. This is a decentralised control protocol for adaptive sampling, designed for an environmental WSN measuring subglacial movement (Glacsweb [Martinez *et al.*, 2004]). In this approach, temporal variations in the environmental parameter being sensed are modelled as a piece–wise linear function, and then the algorithm uses a pre-specified confidence interval parameter in order to make real-time decisions regarding the sampling rate of the sensor nodes. Moreover, linear regression is used to predict the value of future measurements, and if the actual sensor reading exceeds the confidence interval parameter, the sensor starts sampling at an increased rate. However, since the algorithm does not explicitly perform any forward planning, the sensor can rapidly deplete its battery if the increased sampling rate is constantly re-triggered by data that is far from linear.

Furthermore, in tide prediction, Kho *et al.* [2009a] proposed a decentralised algorithm using an information metric that represents the temporal variation in the environmental parameter being sensed. This algorithm, in contrast to USAC, takes energy harvesting into account, and thus, makes the sensors capable of making long-term planning. In particular, this algorithm aims to maximise the information that a sensor collects over a particular time interval subject to energy constraints, and this involves planning exactly when, within the specified time interval, to take a constrained number of samples. The algorithm takes the information provided by the information metric into account when creating a sampling schedule at the beginning of each day. This algorithm also considers the amount of residual energy left in the sensor's battery, and the amount of information that can be collected at different times of the day, based on past experience.

In addition, Jain and Chang [2004] used a similar prediction technique to set the sampling rate adaptively. Their approach employs a *Kalman–filter* (KF) based estimation technique wherein the sensor can use the KF estimation error to adaptively adjust its sampling rate within a given range, autonomously. When the desired sampling rate violates the range, a new sampling rate is requested from a control server. The server allocates new sampling rates under the constraint of available resources such that the KF estimation error over all the active streaming sensors is minimised. However, the main drawback of this technique is that it is centralised, and thus, it is not feasible to operate it in a decentralised setting (see the requirements of our research in section 1.1).

Finally, the algorithm proposed by Osborne *et al.* [2008] uses a multi–output *Gaussian process* (GP) to explicitly model both temporal and spatial correlations between a small number of sensors. The GP is used for adaptive sampling whereby it can determine both the time, and the sensor from which the next sample should be taken, to ensure that the uncertainty regarding the environmental parameter being measured at each sensor location stays below a pre-specified threshold. Thus, here the algorithm only decides when and where the next sampling should be taken. However, the algorithm is centralised, since it requires information from all of the sensors in order to model the spatial correlations between them, and it is relatively computationally expensive.

In this report, we assume that our sampling protocol is a generic adaptive sampling protocol, which can be any of the existing temporal adaptive sampling methods discussed above. We have only one restriction; that is, the algorithm should be decentralised; that is, each agent should be able to autonomously and independently set its sampling rate (for more details see chapter 3). Given this, decentralised techniques, such the sampling protocol of USAC, or the algorithm proposed by Kho *et al.* [2009a], can be used here. On the other hand, due to their centralised manner, the algorithms proposed by Jain and Chang [2004] and by Osborne *et al.* [2008] are not suitable for our model.

### 2.1.2 Information Content Valuation

In order to distinguish important and unimportant data from each other, and thus, to achieve a more efficient information collection in terms of maximising the total information delilvered to the $BS$, an efficient information metric is required to determine the information content of the collected or transmitted data. In our case, this metric is provided by an *information content valuation function*.

Within the tracking literature, where spatially correlated sensor readings typically represent the estimated position of a target, there are a number of standard techniques for defining this function. Most of the works use *Fisher information*, whereby the estimated position of the target is represented as a multidimensional probability distribution, and Fisher information is used to quantify the uncertainty represented by this distribution [Bar-Shalom *et al.*, 2001; Chu *et al.*, 2002; Frieden, 2004; Zhao and Guibasn, 2004]. For example, Chu *et al.* [2002] used acoustic sensors to localise a target. To quantify the information gain of the measured data provided by each sensor, they advocated using the Fisher information matrix as follows. Let $\mathbf{x} = \{\mathbf{x_1}, \mathbf{x_2}, \dots\}$ denote the set of unknown parameters of the target, and $\mathbf{z} = \{z_1, z_2, \dots, z_N\}$ denote the set of sensor measurements, where $N$ is the number of the sensors. Thus, the $ij^{\text{th}}$ component of the Fisher information matrix is:

$$\mathbf{F}_{ij}(\mathbf{x}) = \int p(\mathbf{z}|\mathbf{x}) \frac{\partial}{\partial x_i} \ln p(\mathbf{z}|\mathbf{x}) \frac{\partial}{\partial x_j} \ln p(\mathbf{z}|\mathbf{x}) \, d\mathbf{z}$$

Other approaches were developed by using the Gaussian process. The methods they use generally exploit the fact that the covariance can be analytically evaluated. In more detail, for example, Krause *et al.* [2006] used GPs to model the spatial correlations of locations, in order to determine efficient sensor placements, whereby a maximal information value can be collected by data sampling. In their model, $V$ denotes the set of possible locations, $A$ denotes the set of observable locations and $s$ is an unobservable location. Let $\mathbf{X}_A$ denote the set of observable random variables associated with the locations $A$, and $X_s$ be the random variable associated with location $s$. In order to make predictions at a location $S$ (i.e. to calculate conditional distributions $p\left(X_s = x_s | \mathbf{X}_A = \mathbf{x}_A\right)$), they used the following conditional entropy:

$$H\left(X_s | \mathbf{X}_A\right) = -\int_{x_s, \mathbf{x}_A} p\left(x_s, \mathbf{x}_A\right) \log p\left(x_s | \mathbf{x}_A\right) \, dx_s \, d\mathbf{x}_A$$

Intuitively, this quantity expresses how "peaked" the conditional distribution of $X_s$ given $X_A$ is around the most likely value, averaging over all possible observations $X_A = x_A$ the sensors can make. To quantify how informative the set of data collected from locations $A$ is, they used the criterion of *mutual information* (MI):

$$F\left(A\right) = I\left(X_A, X_{V/A}\right) = H\left(X_{V/A}\right) - H\left(X_{V/A} X_A\right)$$

This criterion expresses the expected reduction of entropy of all locations $V/A$ where sensors were not placed, after taking into account the measurements of sensors at locations in set $A$.

Similarly, Osborne *et al.* [2008] also used GP to model the spacial and temporal correlations and delays between nodes, using Bayesian Monte Carlo techniques to marginalise these hyper-parameters that describe the correlations and delays. They then use the variance of the GPs predictions in order to perform active data selection, which is a decision problem concerning which observations should be taken, deciding when and where to take samples to maintain this variance below a prescribed target level. Their algorithm is computationally efficient as they are learned from the data in an online fashion, and thus, it is capable of performing real-time information processing.

Several other techniques for valuing information include *Shannon entropy*, which is mainly used in signal compression (or coding), target tracking, and information fusion techniques in WSNs [Cover and Thomas, 2006; Hwang *et al.*, 2004]. For instance, Hwang *et al.* [2004] used a belief vector to probabilistically represent the identity of a target. In their work, they considered the problem of combining two belief vectors of the same target from two different sensors (i.e. data fusion). Here, information fusion can be formulated as an optimisation problem such that the fused information is the one that minimises a cost function which represents a performance criterion. This cost function is modelled by the Shannon entropy. More precisely, let $b_1$ and $b_2$ denote the

belief vectors before data fusion, and $b^{'}$ denote the information value of the fused packet. Furthermore, let

$$b^{'} = wb_1 + (1 - w)\, b_2$$

be the fusion strategy. The goal is to determine $w$ such that it minimises the Shannon entropy defined as:

$$H\left(b^{'}\right) = -\sum_{i=1}^{n} b^{'}(i) \log b^{'}(i)$$

where $b^{'}(i)$ denote the probability that the target is in belief state $i$. Informally, Shannon entropy characterises the average amount of information which is gained from a certain set of events. The entropy is maximal when all the events' outcomes are equally likely and, therefore, we are uncertain which event is going to happen. When one of the events has a much higher chance to happen than the others, then the uncertainty (or entropy) decreases. Information value can thus be quantified as the difference between the probabilities of the random event.

In addition, Padhy *et al.* [2006] use the *Kullback–Leibler* (KL) divergence to model the information value of collected data in USAC. Here, KL divergence is a measure of the information gain between a prior and a posterior probability distribution (i.e. the distribution over possible measurements before and after a new item of data has been received). The larger this measure, the less the previous model was capable of explaining the value of the new data, and thus, the more it has to be updated.

More recently, Kho *et al.* [2009a] use the mean Fisher information over a period as a measure that is proportional to the value of information. As a result, when the environment changes at a higher rate, the linear model will start to deviate from the true dynamics of that environment. The value of information will therefore be greater in dynamic situations than in those that are better characterised as static.

In this report, we assume that our model is capable of using any of these techniques, and thus, we do not specify any restrictions on the information valuation technique in use (for more details see chapter 3).

### 2.1.3 Information–Centric Routing

Routing is the process of delivering a message from a source node to a $BS$ inside the same network. A routing algorithm determines actions that a node can use to forward data towards the $BS$, namely (i) transmitting (i.e. which data packets the node should choose to transmit, and to which node), and (ii) receiving (i.e. how many packets a node is required to receive from the other nodes during a transmission period). Since routing is responsible for transporting the data collected by the network to the $BS$, the efficiency of the routing algorithm significantly affects the overall performance of the

network. In fact, routing is one of the most studied areas within the WSN domain, and thus, a large number of algorithms has been proposed for adaptive and efficient data routing in WSNs from different perspectives. These include, but are not limited to, the following: energy efficiency, delay sensitiveness, security, and reliability [Ahdi *et al.*, 2007; Akkaya and Younis, 2005; Al-Karaki and Kamal, 2004; Singh *et al.*, 1998]. However, these algorithms typically do not distinguish important packets from unimportant ones. Thus, this may lead to inefficient performance of information collection, since it may occur that less important data is delivered to the $BS$, while the more important packets are not forwarded at all. Given this, in this section, we focus on routing approaches that are information–centric. In particular, these approaches aim to maximise the total information value delivered to the $BS$. In so doing, they typically use information content valuation techniques in order to determine more important data packets.

One of these algorithms, *directed diffusion* (DD), has been developed by Intanagonwiwat *et al.* [2003]. In DD, the $BS$ sends out a data collection query description by flooding the query to the entire network. That is, data collection happens only when the $BS$ needs a certain type of data. However, since data collection applications (e.g environmental monitoring or area surveillance) typically require continuous data delivery to the $BS$, a significant number of queries will be sent to the network. In this case, the communication cost of DD caused by query floodings is high, meaning DD is not suitable for long–term information collection. To avoid flooding, the *rumor routing* (RR) protocol routes the queries to the nodes that have observed a particular event to retrieve information about the occurrence of the event, and thus, it reduces the total communication cost [Braginsky and Estri, 2002]. However, rumor routing performs well only when the number of events is small. For a large number of events, the algorithm becomes infeasible due to the increase in the cost of maintaining node-event tables in each node. Moreover, all such information–centric protocols do not take into account the network's dynamism (i.e. the changes of network topology) in forwarding data packets.

Apart from the aforementioned approaches, in which information is collected by sending explicit queries from the $BS$, other methods focus on continuous information collection. That is, they provide information collection, without the need of sending any queries, during the whole operation of the network. For instance, USAC (see section 2.1.1), considers the remaining battery power of the communicating nodes and the importance of the data being transmitted, in order to determine the appropriate routing path for the packet. In the similar vein, the *adaptive routing algorithm* (ARA), that has been developed by Zhou and de Roure [2007], beside the battery level and the importance of data, also takes the link cost (assumed to be proportional to the distance) between the nodes into account. However, these protocols are not designed for solving the maximal information throughput routing problem, since their main goal is not to maximise the information throughput between the neighbouring layers of sensor agents, but rather to

identify optimal paths between each node and the $BS$, that can be used for forwarding data.

Finally, an interesting last class of information–centric routing protocols are those that use a market-based control (MBC) paradigm. The use of MBC in WSN allows the use of tools from general equilibrium theory to analyse the behaviour and correctness of a decentralised system. The main market-based protocol includes *self organised routing* (SOR), proposed by Rogers *et al.* [2005], and *self organising resource allocation* (SORA), proposed by Mainland *et al.* [2005]. In more detail, SOR is a mechanism–design based distributed protocol that aims to maximise the network's lifetime. Each node is designed to follow locally selfish strategies which, in turn, result in the self organisation of a routing network with desirable global properties. The protocol consists of a communication protocol, equipping nodes with the ability to find and select a node that is willing to act as a mediator for data relaying, and a payment scheme, whereby a node is rewarded for forwarding messages to the destination. Specifically, the communication scheme identifies potential mediators, the payment scheme allows the sensors to make local selfish decisions which result in good system–wide performance. In contrast, SORA defines a virtual market in which nodes sell goods (e.g. data sampling, data relaying, data listening, or data aggregation) in response to global price information that is established by the end-user. However, this approach again involves an external coordinator to determine the price and it is not clear how this price determination should actually be done in practice. In sum, although these algorithms are based on the multi–agent systems approach, which is clearly related to our model, we do not follow their perspective. In contrast, within this report, we concentrate on networks where sensors maximally cooperate with each other. However, we may take this approach into account in future work.

### 2.1.4 Energy Management

An energy management policy is responsible for allocating energy budgets to sensory tasks, such as sampling and routing data. Most of these policies, however, are typically integrated into the routing part, ignoring the task of sampling. In particular, these methods assume that the data are already sampled, and thus, they focus only in delivering data towards the $BS$. Furthermore, they typically follow the concept of energy–awareness; that is, they aim to minimise the energy consumption of each node, while data forwarding is still to be done.

Given this, a number of energy–aware algorithms use clustering techniques to minimise energy consumption in sensor networks through the rotation of cluster–heads such that the high energy consumption in communicating with the $BS$ is spread across all nodes. These algorithms include *low energy adaptive clustering hierarchy* (LEACH), proposed

by Heinzelman *et al.* [2000], and *power efficient gathering in sensor information systems* (PEGASIS), proposed by Lindsey and Raghavendra [2002]. In general, these methods make good attempts to try to balance the energy consumption by electing cluster–heads, each of which is responsible for relaying the data from a subset of nodes back to the $BS$ in an intelligent way. However, these cluster–heads all need to be placed inside the $BS$'s radio range as they communicate with it directly. Thus, this assumption limits the size of the monitoring environment, since the wireless radio range of the $BS$ is limited. Moreover, these single cluster–heads can become a communication bottleneck of the network, since in each round the cluster–heads need to communicate with a large number of nodes within their cluster. Hence, this aspect contains some of the drawbacks of the centralised control regime.

The life span of the network can also be lengthened by reducing the total energy consumption needed to deliver the packets to the $BS$. From this perspective, Dekorsy *et al.* [2007] proposed an approach that jointly controls the routing and energy management, in order to achieve efficient data forwarding. In particular, the approach aims to minimise the total energy consumption of each node, while the collected data has to be delivered to the $BS$ using multipath routing (i.e. there can be multiple routing paths between a node and the $BS$). In so doing, the approach considers each node's residual energy level, the transmission power level, and maximal communication bandwidth. This approach, however, assumes that the data is already sampled, and that future data is not taken into consideration when optimal routing paths are calculated. Given this, whenever the environment changes, it has to recalculate the optimal paths, and thus, it requires significant computational resources.

In addition, another way to to lengthen the life span of the network is to perform *energy balancing* [Dinga *et al.*, 2004]. That is, to maximise the residual energy level of the bottleneck node (i.e. the node with the least energy level) in the network during the routing. In this vein, Ok *et al.* [2009] used a metric to take the energy cost of transmission, as well as the sensors' remaining energies into account. This metric gives rise to the design of the *distributed energy balanced routing* (DEBR) algorithm, to balance the data traffic of sensor networks in a decentralised manner. Furthermore, Li *et al.* [2007] proposed a global–energy balancing routing scheme (GEBR) for real–time traffic. Now, while both of these algorithms perform well in prolonging the lifetime of the WSN, they are not designed for adapting to environmental changes, since they do not take these changes into account.

More recently, Merrett [2008] developed the *information managed energy aware algorithm for sensor networks* (IDEALS) protocol, which is aimed for extending the network lifetime of WSNs. IDEALS is an application specific heuristic protocol as it requires that every sensor node decides its individual network involvement based on its own energy state and the importance of information contained in each message. Similarly, USAC

uses an *opportunity cost* of the energy used by each sensor to balance the energy consumption of the tasks of sampling and forwarding. That is, by evaluating its own opportunity cost, each sensor can decide whether it spends energy on sampling or forwarding, depending on which is the more preferable opportunity for the sensor. Moreover, USAC also considers the total energy consumption required to transmit a packet along a particular path as well. These methods, since they can vary the energy budgets allocated to the sensory tasks, are suitable for adapting to environmental changes. However, they do not take energy neutrality into account. Given this, an increasing percentage of sensor batteries will deplete as time goes by, and thus, the global performance of the network will be decreased in the long term.

To date, very few studies have focused particularly on energy management in networks of energy harvesting nodes. A notable exception, however, is the work of Kansal and Srivastava [2003]. In their work, the authors used spectral estimation functions and prediction filters to estimate the expected amount of harvested energy of each sensor in a given future time interval. Based on these estimations, the sensors are able to schedule their tasks, in order to achieve long-term goals. However, their approach does not focus on information-centric routing, and thus, will not perform well in information collection. More recently, Kho *et al.* [2009b] proposed an energy neutral information-centric data collection algorithm, that combines sampling and routing policies, in order to maximise the collected information in one time slot. That method, however, does not plan for long-term operation, since it is optimised for one–slot time intervals. Furthermore, it does not consider dynamic changes of the environment.

## 2.2 Reinforcement Learning

In this report, we aim to use a learning theory approach to achieve long-term performance in information collection of WSNs. In this context, agents will be able to learn efficient policies and optimal behaviours in an unknown domain, without any (or with minimal) prior knowledge of that domain. Now, since agents are required to operate in an unsupervised manner, we focus on the approach of reinforcement learning, which is an efficient and robust approach designed to tackle such learning problems. Given this, in this section, we briefly review the standard reinforcement learning framework of discrete time, called the finite *Markov decision process* (MDP), which underlies our work.

In the MDP framework, a learning agent interacts with an environment at some discrete time scale, $t = 0, 1, 2, \ldots$. On each time step $t$, the agent perceives the state of the environment $s_t \in S$, where $S$ is the set of possible states of the environment. On that basis, the agent chooses a primitive action $a_t \in A_{s_t}$. In response to each action $a_t$, the

environment produces one step later a numerical reward $r_{(t+1)}$, and a next state $s_{(t+1)}$. For the sake of simplicity, we let $A = \cup_{s \in S} A_s$ denote the union of the action sets.

At each time step, the agent implements a mapping from states to probabilities of selecting each possible action. This mapping is called the agent's *policy* and is denoted with $\pi_t$ , where $\pi_t(s, a)$ is the probability that $a_t = a$ if $s_t = s$. Reinforcement learning methods specify how the agent changes its policy as a result of its experience. The agent's goal, roughly speaking, is to maximise the total amount of reward it receives over the long run. More precisely, let $T$ denote the operation time of the agent, and $t$ the current time, respectively. Thus, the goal is to maximise the following:

$$R_t = \sum_{k=0}^{T} \gamma^k r_{t+k+1} \tag{2.1}$$

where $\gamma$ is the discount coefficient. The discount coefficient determines the present value of future rewards; a reward received $k$ time steps in the future is worth only $\gamma^{k-1}$ times what it would be worth if it were received immediately. If $\gamma = 0$, the agent is *myopic* in being concerned only with maximising immediate rewards; its objective in this case is to learn how to choose $a_t$ so as to maximise only $r_{t+1}$. If each of the agent's actions happened to influence only the immediate reward, not future rewards as well, then a myopic agent could maximise equation 2.1 by separately maximising each immediate reward. But, in general, acting to maximize immediate reward can reduce access to future rewards so that the return may actually be reduced. Thus, as $\gamma$ approaches 1, the objective takes future rewards into account more strongly: the agent becomes more farsighted.

Given this, now we focus on the process of maximising equation 2.1 in more detail. Assuming that $S$ and $A$ are finite, the environments transition dynamics given state $s$ and action $a$ can be modeled by one-step state-transition probabilities as follows.

$$\begin{aligned} p^a_{ss'} &= Pr\{s_{t+1} = s | s_t = s, a_t = a, s_{t-1}, a_{t-1}, s_{t-2}, a_{t-2}, \dots\} \\ &= Pr\{s_{t+1} = s | s_t = s, a_t = a\} \end{aligned} \tag{2.2}$$

for all $s, s' \in S$ and $a \in A_s$. That is, $p^a_{ss'}$, which denotes the probability that from state $s$, with action $a$, the agent will perceive state $s'$, depends only on the current state and the current action, and does not depend from the past states and actions (Markov property). Thus, the one-step expected rewards is:

$$r^a_s = \mathbf{E}\{r_{t+1} | s_t = s, a_t = a\} \tag{2.3}$$

for all $s \in S$ and $a \in A_s$. The two sets of quantities defined in equations 2.2 and 2.3 together constitute the one-step model of the environment.

To achieve the long-run objective given in equation 2.1, the agent has to choose the policy that maximises the expected discounted future reward from the current state $s$. Let $V^{\pi}(s)$ denote the expected discounted future reward of policy $\pi$ from state $s$. That is,

$$\begin{aligned} V^{\pi}(s) &= \mathbf{E}\{r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \ldots | s_t = s, \pi\} \\ &= \mathbf{E}\{r_{t+1} + \gamma V^{\pi}(s_{t+1}) | s_t = s, \pi\} \\ &= \sum_{a \in A_s} \pi(s,a) \left[ r_s^a + \gamma \sum_{s'} p_{ss'}^a V^{\pi}(s') \right] \end{aligned} \tag{2.4}$$

The quantity $V^{\pi}(s)$ is called the *value* of state $s$ under policy $\pi$, and $V^{\pi}$ is called the *state-value* function for $\pi$. The optimal state-value function gives the value of each state under an optimal policy is:

$$\begin{aligned} V^*(s) &= \max_{\pi} V^{\pi}(s) = \max_{a \in A_s} \mathbf{E}\{r_{t+1} + \gamma V^{\pi}(s_{t+1}) | s_t = s, \pi\} \\ &= \max_{a \in A_s} \left[ r_s^a + \gamma \sum_{s'} p_{ss'}^a V^*(s') \right] \end{aligned} \tag{2.5}$$

Any policy that achieves the maximum in equation 2.5 is by definition an optimal policy. Thus, given $V^*$, an optimal policy is easily formed by choosing in each state $s$ any action that achieves the maximum in equation 2.5. Planning in reinforcement learning refers to the use of models of the environment to compute value functions and thereby to optimise or improve policies.

Another way to approach the maximal value of equation 2.1 is to determine the *action-value function* for policy $\pi$. That is, the value of taking action $a$ in state $s$ under policy $\pi$, denoted $Q^{\pi}(s,a)$, is the expected discounted future reward starting in $s$, taking $a$, and henceforth following $\pi$. This value can be evaluated as follows.

$$\begin{aligned} Q^{\pi}(s,a) &= \mathbf{E}\{r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \ldots | s_t = s, a_t = a, \pi\} \\ &= r_{t+1} + \gamma \sum V^{\pi}(s_{t+1}) \\ &= r_s^a + \gamma \sum_{s'} p_{ss'}^a \sum_{a'} \pi(s',a') Q^{\pi}(s',a') \end{aligned} \tag{2.6}$$

Thus, the optimal action-value function can be evaluated as follows.

$$Q^*(s,a) = \max_{\pi} Q^{\pi}(s,a) = r_s^a + \gamma \sum_{s'} p_{ss'}^a \max_{a'} Q^*(s') \tag{2.7}$$

There are a number of efficient technique to calculate the optimal action-value function $Q^*(s,a)$, or the optimal state-value function $V^*(s)$, such as temporal-difference, Q-learning or dynamic programming methods. The description of these algorithms can be

found in any survey work of the area, such as in Sutton and Barto [1998] or in Kaelbling *et al.* [1996].

To date, reinforcement learning has been widely used in a number of areas, mostly related to the domain of artificial intelligence. For example, reinforcement learning has been used in autonomous helicopter flight control [Abbeel *et al.*, 2006], robot football coordination [Duan *et al.*, 2007], robotic-arm control [Schaal and Atkeson, 1994], and computer games as well [Littman, 1994]. To fit the problems in these domains, beside the standard MDP, that is previously described, researchers has studied a number of other related MDP models. These include, but are not limited to, the following: partially observable MDPs (POMDPs), decentralised POMDPs (DecPOMDPs), and multi–armed bandits (MAB). In particular, POMDPs deal with the case when the state of the environment is not always fully observable (e.g. the robot cannot see all the changes in its environment). Thus, in this case, the standard MDP model is extended with the set of observable events [Cassandra, 1998]. In addition, DecPOMDP is an extension of POMDP, which deals with multi–agent learning, in which the agents together have to determine the optimal *joint action set* (i.e. the set of each agent's action). In DecPOMDPs, the decisions are made in a decentralised way; that is, each agent autonomously decides its action, cooperating with the others [Seuken and Zilberstein, 2008]. In contrast, MAB is a simplified version of the MDP model. In the MAB model, the state of the environment is ignored. Thus, MAB can be regarded as an one–state MDP model, which makes it the simplest RL approach [Gittins, 1989; Lai and Robbins, 1985; Robbins, 1952]. As mentioned in section 1.2, in this report, we focus on the use of MAB models, due to its simplicity compared to other RL models. This decision is reasonable, since one of our research objectives is to provide information collection approaches with low computational cost. Given this, we describe the MAB model in more details in the following section.

## 2.3 Multi–Armed Bandit Problems

In this section, we first describe the standard MAB model in detail. Then we discuss the application of MAB in different research domains, demonstrating the usefulness of MAB. Following that, we focus on the introduction of other related MAB models, that contain several modifications, compared to the standard MAB.

In more detail, the standard *multi–armed bandit* (MAB) model can be described as follows. It was originally proposed in Robbins [1952]. In the MAB problem, there is a machine with $K$ arms, each of which delivers rewards, that are independently drawn from an unknown distribution, when the machine's arm is pulled. Given this, a gambler must choose which of these machines to play. At each time step, he pulls the arm of one

of the machines and receives a reward or payoff. The gambler's purpose is to maximise his return, that is, the sum of the rewards he receives over a sequence of pulls. As the reward distributions differ from arm to arm, the goal is to find the arm with the highest expected payoff as early as possible, and then to keep gambling using that best arm.

Using the terminology of multi–agent systems hereafter, we refer to the gambler as an agent, and refer to each arm pulling action of the gambler as an action of that particular agent. Thus, we can formulate the MAB problem as follows. Let $K$ denote the number of the actions that the agent can make. At each time step $t$, the agent makes action $a_t$, which delivers the reward $r_{a_t}(t)$. Finally, let $T > 0$ denote the time horizon in which the agent operates. Thus, we have the following optimisation problem:

$$\max \sum_{t=1}^{T} r_{a_t}(t) \tag{2.8}$$

Thus, the agent has to choose a policy, that is, a sequence of actions, that may deliver the maximal reward at each time step $t$ in order to achieve the maximum of equation 2.8. According to this, the MAB problem can be regarded as a special instance of a reinforcement learning (RL) problem which is a single state Markov decision process (MDP) (for more details see section 2.2).

A fundamental dilemma in the MAB problem is the trade-off between exploration and exploitation. On the one hand, if the agent exclusively chooses the action that he thinks is the best (i.e. exploitation), he may fail to discover that one of the other actions actually has a higher expected payoff. On the other hand, if he spends too much time trying out all the actions and gathering statistics (i.e. exploration), he may fail to choose the best action often enough to get a high return.

Against this background, researchers have proposed many approaches that tackle this exploration-exploitation conflict from different aspects. In these works, the agent's performance is typically measured in terms of *regret*, which is defined as the difference between the expected return of the optimal strategy (consistently performing the current best action) and the agent's expected return. Given this, by using statistical assumptions, Lai and Robbins [1985] proved that the agent's regret over $T$ rounds can be asymptotically as small as $O(\ln T)$. That is, for $T \to \infty$, the regret converges to $O(\ln T)$. Furthermore, they proved that this bound is optimal in the following sense: there does not exist a policy for the agent with a better asymptotic performance. More recently, Agrawal [1995] introduced a family of policies which are much easier to compute, compared to those proposed by Lai and Robbins [1985]. In addition, policies that achieve logarithmic regrets uniformly over time are proposed in Auer *et al.* [2002]. Such policies, for example, are $\epsilon_n$–greedy, and *upper confidence bound* (UCB) algorithms. In the former algorithm, at each step $n$, the arm with the best current estimate (i.e. the

experimental average reward value) is pulled with probability $(1 - \epsilon_n)$, while a different arm is randomly pulled with probability $\epsilon_n$. That is, the algorithm exploits with probability $(1 - \epsilon_n)$, and explores with probability $\epsilon_n$ at each step $n$. Here, the value $\epsilon_n$ decreases as $n$ grows; that is, as time goes by, the algorithm focus more on exploitation, while it decreases the probability of exploration [Auer *et al.*, 2002; Sutton and Barto, 1998]. On the other hand, the latter algorithm maintains an upper confidence bound estimate for each arm's real mean reward value. At each time step, the algorithm then chooses the arm with the highest upper confidence bound to pull, and updates the new value of the bound [Auer *et al.*, 2002].

MAB models, due to its efficiency in handling the trade–off between exploration and exploitation, have been used in a variety of areas. Such areas include clinical trials where different treatments need to be experimented with, while patient loss should also be minimised as well [Hardwick and Stout, 1991]. MAB models are also used to solve financial and investment problems as well. For example, optimal, but a priori unknown investment options can be learned by using MAB exploration, while income maximisation is provided by MAB exploitation [Lai and Lim, 2005; Wang and Wang, 2009]. In addition, MAB can also be adopted to the area of e–commerce, as an efficient way to identify the ranking of web documents [Radlinski *et al.*, 2008], or as a learning technique for optimising online advertisement [Chakrabarti *et al.*, 2008].

Although the standard MAB model allows an elegant treatment of the exploration-exploitation trade-off in the aforementioned domains, it gives an incomplete description of the sequential decision making problem facing an agent in many other real world scenarios. Specifically, in the standard MAB, one of the fundamental assumptions is that the reward value of each action can be denoted with a stationarily distributed random variable. However, the stationarity of the distributions is not always the right assumption in a number of cases. For example, in the case of an agent within a dynamically changing environment, the expected value of the rewards are time dependent, that is, non–stationary. Thus, the aforementioned approaches may not perform efficiently in such cases, since there is no fixed expected value for the reward of each action. In addition, making an action (i.e. pulling an arm) does not cost a thing in the standard MAB. However, in many applications, making an action typically consumes a certain amount of limited resources as well, and thus, arm pulls are costly.

In the WSN domain, since the environment is typically dynamic, standard MAB models are not suitable for modelling the energy management problem (see chapter 5 for more detail). Furthermore, in the case of energy management with non–harvesting sensors, each arm pull is costly as well (see chapter 6 for more detail). Against this background, in the following sections, we provide an overview of MABs with non–stationary environments and costly arms, which are related to MAB models we use in this report for dealing with the energy management problem. Specifically, section 2.3.1 introduces the

non–stationary varieties of MAB. In addition, it describes an efficient MAB model, the non–stochastic MAB in more details. Following this, section 2.3.2 discusses MAB models that consider pulling cost constraints.

### 2.3.1 Non–Stochastic Multi–Armed Bandits

In order to deal with non–stationary MAB, researchers have proposed a number of approaches. For example, Koulouriotis and Xanthopoulos [2008] used several RL techniques and evolutionary algorithms to handle the non–stationarity in the MAB model. In addition, for piecewise–stationary (i.e. the environment is regarded as stationary in periods of time) MAB, Yu and Mannor [2009] developed an efficient method and provided tight performance bounds for the method. However, these approaches assume that certain statistical knowledge about the environment is provided a priori. In particular, the work of Yu and Mannor assumes that the environment is stationary in subsequent intervals of time, and thus, within an interval, it is regarded as stationary. On the other hand, the algorithms proposed by Koulouriotis and Xanthopoulos are only proposed for a certain class of statistical distributions (e.g. Gaussian). These assumptions, however, are not suitable for many MAB environments, and thus, the aforementioned approaches will not perform well in such situations.

Against this background, to address this shortcoming, a *non–stochastic multi–armed bandit problem* has been introduced, in which *no statistical assumptions* are made about the generation of rewards of each action [Auer *et al.*, 2003; Gittins, 1989; Ishikida and Varaiya, 1994]. In this case, each action is initially assigned an *arbitrary and unknown* sequence of rewards, one for each time step, chosen from a bounded real interval. At each time step, the agent performs an action and receives the corresponding reward from the sequence assigned to that chosen action. The rest of the sequence, however, remains unrevealed for the agent. This aspect makes the model of reward values more generic and thus, easier to handle. To solve the non–stochastic MAB problem, Auer *et al.* [2003] have proposed an algorithm, named Exp3, which achieves efficient performance in dynamic environments (see section 5.1.1 for more details).

The aforementioned relaxation on the statistics of the rewards, however, significantly reduces the accuracy of the algorithms proposed for the MAB problem [Auer *et al.*, 2003]. That is, they produce worse bounds for the regret, compared to the theoretical optimum of $O(\ln T)$ that can be achieved by statistical model based algorithms. In particular, Auer *et al.* [2003] have proved that the regret in the non-stochastic MAB problem has a lower bound of $O\left(\sqrt{T}\right)$. Despite this drawback, the Exp3 algorithm is still useful in a number of cases, since it can provide acceptably small bounds of the regret, for any of the time horizon $T$ (see section 5.1.2 for more details). This property is more desirable when the operation time of the agents are small, or when good, but

not necessarily optimal performance is required from the beginning (compared to the property of asymptotic convergence that statistical model based algorithms could provide [Kleinberg, 2006]).

Apart from the Exp3 algorithm for the non–stochastic MAB model, another commonly used approach for non–stationary MAB problems is the $\epsilon$–greedy policy [Sutton and Barto, 1998]. This policy is the simple version of the $\epsilon_n$–greedy. Here, the value of $\epsilon$ is fixed over time, and thus, the probability of exploration is not decreased as time goes by. Due to this reason, the algorithm is always able to re–explore, and thus, it can learn the changes of the environment. This algorithm, however, does not have theoretical bounds for its performance, such as in the case of the Exp3 algorithm.

### 2.3.2 Multi–Armed Bandits with Cost Constraints

Another way to extend the standard MAB is to take costly arms into account. To date, a number of researchers have focused on MABs with a limited exploration budget. Specifically, they consider cases where pulling an arm incurs a pulling cost, whose currency is not commensurable with that of their rewards. The agent's exploration budget then limits the number of times it can sample the arms, thus defining an initial exploration phase, during which the agent's sole goal is to determine the optimal arm to pull during the subsequent cost free exploitation phase. This extension is motivated by the fact that in many applications (e.g data mining or clinical trials) searching for the best option (exploration) is typically costly (e.g. it requires a lot of time, or money) [Madani *et al.*, 2004]. In such cases, the exploration phase in the MAB model is limited. Another motivation is from the problem of maximising some function $f$, observed with noise, whereby evaluating $f$ at a point is costly (e.g., in terms of numerical or financial costs). The issue is to choose as adequately as possible where to query the value of this function in order to have a good approximation to the maximum [Bubeck *et al.*, 2009b; Kleinberg, 2005].

Given this, several exploration policies have been developed that maximise the long–run expected reward of an agent faced with this type of limit on exploration, namely: (i) MAB with pure exploration [Bubeck *et al.*, 2009a]; (ii) budgeted MAB [Guha and Munagala, 2007]; and (iii) MAB with max–loss value–estimation [Antos *et al.*, 2008]. In particular, Bubeck *et al.* proposed efficient *anytime* methods for pure exploration in limited, but unknown time interval. More precisely, in this work, the authors analysed uniform (i.e. the arms are subsequently and uniformly pulled) and UCB–based (i.e. the next arm to pull is determined by the UCB) exploration policies. These methods are anytime, since the more time they have to run, the better the result they can provide. In the contrast, Guha and Munagala devised approximation algorithms for this MAB with limited exploration problem (which they denote as budgeted MAB). Finally, Antos *et al.* studied an interesting problem, in which the goal minimise the uncertainty of value

estimates of each arm (i.e. the difference between the arm's real reward value mean and its estimate).

## 2.4 Learning in Wireless Sensor Networks

To date, very few approaches have attempted to use reinforcement learning in the WSN domain. Notable exceptions, however, typically focus on adaptive routing. In this research area, a number of researchers proposed learning-based techniques to maintain a routing tree that efficiently handles the node failures, creating an efficient connectivity between the nodes to the $BS$ in any circumstances [Zhang and Huang, 2006]. In addition, Aghaeil *et al.* [2007] developed swarm intelligence-based algorithms which use reinforcement learning for data packet routing in WSNs. Furthermore, Gelenbe and Lent [2004] proposed a concept called a *Cognitive Packet Network* (CPN) for intelligent packet forwarding in wireless ad hoc networks. In particular, CPN is an autonomous adaptive *quality of service* (QoS) driven network, which adaptively selects paths so as to offer best effort QoS to the end users based on user defined QoS. CPN uses neural network based reinforcement learning to make routing decisions separately at each node. These approaches, however, do not focus on maximising the collected information in a long-term operation time, and thus, may not perform well in long run.

Apart from routing, a few works have studied areas such as *sensor localisation* (i.e. sensor placement in the monitored environment), efficient *medium access control* (MAC) protocol design or adaptive resource/task management of WSNs. In more detail, Seah *et al.* [2007] focused on the efficient spatial coverage of the monitored area, using learning techniques. Meanwhile, Liu and Elhanany [2006] proposed a reinforcement learning based MAC protocol for WSNs. In their protocol, nodes actively infer the state of other nodes, using a reinforcement learning based control mechanism, thereby achieving high throughput and low power consumption for a wide range of traffic conditions. Finally, and mostly related to this work, Shah and Kumar [2007] advocated the use of the reinforcement learning approach to achieve efficient task management in WSNs. In their work, at each time step, each agent can independently perform one of the following tasks: transmit, receive, sample, alarm, and aggregate (i.e. merge packets with similar content to each other, compressing their total size). Thus, the goal is to allocate the tasks to the agent efficiently so that a global objective is achieved. The former two topics are outside the scope of our interest, since we do not either focus on localisation or MAC protocol design. The last work, in fact, shows similarities to our work. However, since it uses a complex RL model, its computational cost is significant. Furthermore, that approach is not designed for information–centric routing, and thus, it will not perform well in long–term information collection.

## 2.5 Summary

In this chapter, we have reviewed the literature related to information collection in the WSN domain. We divided the state-of-the-art approaches into four groups, based on the perspective from which they approached the data collection problem. These perspectives are the following: adaptive sampling, information content valuation, information–centric routing, and energy management. Given this, we first discussed the most common adaptive sampling algorithms, some of which were centralised, while the others followed the decentralised paradigm. We pointed out that in our model, any of decentralised algorithms can be used. Following this, we reviewed a number of information content valuation methods, which play an important role in our model, as they can distinguish important from unimportant data. Then we compared and contrasted a number of state-of-the-art information–centric routing algorithms. We have shown that they all suffer from at least one of the following limitations:

- They are not adaptive to the environmental changes; that is, they do not satisfy one of our research requirements.
- They are not designed for solving the maximal information throughput routing problem.
- The nodes do not cooperate with each other in order to maximise the amount of collected information.

Furthermore, we have discussed energy management techniques, including energy consumption minimisation, energy balancing, and energy harvesting. The former two approaches follow the concept of energy–awareness, and thus, will not perform well in the long–term. Meanwhile, the latter is a key feature of our model that allows long–term information collection for our agents.

In the second part of this chapter, we have described the MDP model, which is a standard reinforcement learning framework of discrete time. Then we focused on a simplified version of the RL, namely the MAB problem domain. In particular, we discussed the non–stochastic MAB problems, that can handle non–stationary environments. Furthermore, we also discussed MAB models with cost constraints. Following this, we reviewed the existing RL applications in the WSN domain, showing that only very few works have attempted to use this approach in the domain.

In sum, we have shown that a number of studies have been undertaken in order to achieve efficient information collection in the WSN domain. Specifically, we have demonstrated that, to date, none of the state-of-the-art studies has designed such a mechanism for long–term information collection, which satisfied all of our research requirements, namely: (i)

adaptivity; (ii) scalability; (iii) efficient computational complexity; and (iv) low communication cost.

Against this background, one of the main drives of our research work is to fill this gap, by designing adaptive behaviour for each of the agents in the network, in order to achieve efficient information collection in the long run. Furthermore, we aim to meet the research requirements. In so doing, we focus on developing energy management and information–centric routing techniques, while we use state–of–the–art sampling and information content valuation techniques in our model.

To this end, in chapter 3, we introduce a generic model for WSNs that includes all of the aforementioned features such as sampling, routing, energy management and learning. Following that, in chapter 4, we propose an information–centric routing algorithm, that provides an optimal result for the maximal information throughput routing problem. Then we introduce a multi–armed bandit based algorithm that is responsible for efficient energy management for energy–harvesting sensors in chapter 5. Finally, we propose a MAB learning based method for energy management with non–harvesting sensors, which is based on a new, budget limited, MAB model (see chapter 6 for more details).

# Chapter 3

# Formal Models and Problem Definitions

In this chapter we present a formalisation of the long–term information collection problem for WSNs. To this end, we first provide a formal description of the WSN system in section 3.1. In particular, we describe the models of adaptive sampling, information content valuation, data routing, and energy management policies that play fundamental roles in efficient information collection of WSNs. Here, we also discuss the assumptions, on which the model formalisation is based. Following this, in section 3.2, we formulate the main objective of our research: that is, to achieve efficient long-term information collection in WSNs. Finally, we introduce two problems we aim to solve to achieve this goal, namely (i) the energy management problem; and (ii) the maximal information throughput routing problem, in sections 3.3, and 3.4, respectively.

## 3.1 Wireless Sensor Network Model

In this section, we introduce our WSN model, covering the energy management, sampling, information content valuation, and routing parts, respectively. In so doing, we first simplify the complexity of the real world WSN model, by making the following assumptions about the physical world of the network:

- We make no assumptions regarding the underlying radio propagation model of the WSN. We also assume that the communication between the nodes is *perfect* (i.e. no data loss occurs).
- In our model, *time synchronisation* is *perfect*. In real world WSN applications, in order to decrease the energy consumption of each node, the communication module

of the node is turned on and off periodically, thus, forming active and sleep modes, respectively. In a decentralised manner, however, it may happen that the sending node is already active, but the receiving node is still in sleep mode, and thus, data loss may occur in this situation. To minimise the number of such situations, time synchronisation policies are used in WSNs [Elson and Estrin, 2001; Sundararaman *et al.*, 2005]. These policies synchronise the internal clocks of the nodes to each other so that they can schedule their mode changes synchronously. Here, we assume that such aforementioned situations cannot happen during the operation of the WSN.

- We also assume that *interference does not occur*. Interference occurs when multiple nodes try to use the same communication channel at the same time, causing data collision, such that none of the communication data can be detected at the receiving side. Such collisions are typically avoided through the use of a *medium access control* (MAC) protocol that controls the data transmission of each node [Demirkol *et al.*, 2006; Wu and Biswas, 2007]. Thus, we assume that the MAC protocol we are using here can avoid data collision during transmission.

- The WSN that we are studying is not a mobile network (i.e. the sensors cannot change their location), however, link failures, node failures and node additions are taken into account. That is, the network can be *topologically dynamic, but not mobile*.

- In our model, the sensors have *no memory limitations*. That is, the collected and received data can be stored in each sensor without deletion. Moreover, in our model, the *energy consumption of memory management* (i.e. reading from memory and writing to memory) *is negligible* compared to the energy consumption of data sampling and forwarding. This assumption is reasonable according to the experimental studies reported in Mathur *et al.* [2006] and Anastasi *et al.* [2004].

- Finally, each node can *recharge its battery periodically*, making it independent of human intervention.

Based on these assumptions, we can formulate the WSN model as follows. Let $I = 1, 2, \ldots, N$ be the set of agents in the network, which contains one base station, denoted $BS$[1]. We assume that each agent knows its distance in hops from the $BS$. This can be achieved by using any of the standard shortest path algorithms (e.g. distributed breadth-first search or distributed Bellman–Ford). Furthermore, each agent can only communicate with those who are inside its communication range. Different agents may have different sizes of range.

[1] Our model can easily be extended to cover systems with multiple base stations.

Here, for the sake of simplicity, we split the time line into slots. That is, hereafter we assume that time is discrete, and can be denoted with the sequence of $t = 0, 1, 2, \ldots$. We consider three specific kinds of energy consumption for each agent in the network, namely: the energy required to (i) acquire (i.e. sample), $e_i^{\mathrm{S}}$; (ii) transmit, $e_i^{\mathrm{Tx}}$; and (iii) receive, $e_i^{\mathrm{Rx}}$, a single data packet (we assume that each packet has the same size in bytes). Besides, let $B_i(t)$ denote the amount of energy budget agent $i$ can use each slot $t$. For energy–harvesting sensor agents, $B_i(t)$ is equal to the amount of harvested energy at slot $t$, since the agents comply the concept of energy–neutrality. On the other hand, when energy–harvesting is not possible, then $B_i(t)$ is equal to the total residual energy level of the agent. In addition, we disregard the energy required for other types of processing since it is negligible in comparison [Mathur *et al.*, 2006; Merrett, 2008].

For data sampling, we do not place particular restrictions on the techniques used here, any existing adaptive technique can be employed. Examples of such techniques can be found in section 2.1.1.

To calculate the importance of sampled data, we use information content valuation methods. Similar to the sampling case, any existing technique from the literature can be used for this (see 2.1.2 for more details). Furthermore, we also assume that the information value of the collected data is *discounted* over time by a factor $\lambda \in (0, 1]$ (i.e. loses its value as time passes by), if it is not delivered to the $BS$ yet. This assumption is justified by the fact that in many applications, new information is more preferable older information.

In existing routing protocols, agents typically forward data to other agents, which are closer to the $BS$, either in terms of physical distance or number of hops. Thus, following this concept, we assume that in our model, agents can send data to those which are closer to $BS$ in terms of number of hops. Finally, we assume that data sampled or received at each agent $i$ at slot $t$ can only be forwarded from slot $(t + 1)$. This assumption is also reasonable, since without it, newly sampled data could be delivered to the $BS$ instantaneously.

## 3.2 The Long–Term Information Collection Problem

In the previous section, we have introduced the model that considers adaptive sampling, routing, information valuation and energy management of WSNs. Given this, we now aim to give a formal description of the research objective, that is, to maximise the total collected information in WSNs, in a given finite time interval.

In more detail, let $\mathbf{S}_i(t)$, $\mathbf{Rx}_i(t)$ and $\mathbf{Tx}_i(t)$ denote the set of sampled, received and transmitted data packets of agent $i$ at round $t$. Let $p$ denote a single data packet, whose

information value at round $t$ is $v(p,t)$. Furthermore, we assume that the WSN operates in the finite time interval $[0,T]$[1]. Given this, our objective is formulated as follows:

$$\max \sum_{t=0}^{T} \left\{ \sum_{p \in Rx_{BS}(t)} v(p,t) \right\} \tag{3.1}$$

That is, we aim to maximise the total information value delivered to the $BS$ over the time interval $[0,T]$, with respect to the following constraints:

$$\mathbf{Tx}_i(t) \subseteq \mathbf{Q}_i(t) \tag{3.2}$$

for each agent $i$ and round $t$, where $\mathbf{Q}_i(t)$ is the set of total transmittable data packets in the memory. That is, the transmitted data is the subset of the total transmittable data (packets that were sampled or arrived until the previous round) of each agent $i$. Furthermore,

$$\mathbf{Q}_i(t+1) = (\mathbf{Q}_i(t) / \mathbf{Tx}_i(t)) \cup \mathbf{S}_i(t) \cup \mathbf{Rx}_i(t) \tag{3.3}$$

for each agent $i$. That is, the set of transmittable data of agent $i$ at round $(t+1)$ is the union of the sets of residual data (i.e. $(\mathbf{Q}_i(t) / \mathbf{Tx}_i(t))$), the received data and the sampled data at round $t$.

For the concept of energy-neutrality, we have the following constraints:

$$e_i^{\mathrm{S}} |\mathbf{S}_i(t)| + e_i^{\mathrm{Rx}} |\mathbf{Rx}_i(t)| + e_i^{\mathrm{Tx}} |\mathbf{Tx}_i(t)| \leq B_i(t) \tag{3.4}$$

for each agent $i$, where $|\{.\}|$ denotes the size of set $\{.\}$. This constraint demonstrates that the energy consumption of each action made by agent $i$ cannot exceed the energy budget given in round $t$.

Furthermore, for each $p \in \mathbf{S}_i(k) \cup \mathbf{Rx}_i(t)$ (i.e. received data or sampled data of agent $i$ at round $t$), that is not delivered to the $BS$ before time slot $t$:

$$v(p,t+1) = \lambda v(p,t) \tag{3.5}$$

where $\lambda \in (0,1]$ is the discount coefficient. That is, the information value of packet $p$ is decayed with the discount factor $\lambda$, as time goes by.

As mentioned in section 1.2, to efficiently solve the problem formulated in equation 3.1, we separately study the energy management and routing of the WSN, while we assume that efficient sampling and information content valuation can be achieved by

[1]In the case of infinite time interval, since the energy harvesting is periodical, it may occur that the agents are able to operate within that infinite time. Thus, the amount of information value collected at the base stations can be infinite as well. In this case, a different objective should be introduced, instead of maximising the amount of collected information value. This, however, remains as future work.

using existing techniques. Given this, section 3.3 discusses the energy management problem in more detail, while section 3.4 focuses on the routing problem.

## 3.3 The Energy Management Problem

The definition of the energy management problem is based on the observation that since each agent can sample, receive or transmit data, it is necessary for the agents to vary the energy budget they associate with each of these action types, so that their overall performance can effectively adapt to environmental changes. That is, by adaptively setting the value of the energy budgets, the agents can decide whether to put more effort on sampling (e.g. when significant events are occurring in the monitored area), receiving important data from the others (e.g. when they have collected high value information that has to be delivered to the $BS$), or transmitting data (e.g. when the delivery of data cannot be delayed too long). With such capabilities, our hypothesis is that the agents should achieve better performance than systems without the ability to adapt in this fashion. However, in order to determine the optimal budget settings (*exploitation*), the agent first has to learn the performance of other setting combinations as well (*exploration*). Furthermore, due to the environmental changes, the agent has to repeatedly re-learn the current optimal settings, since it may also vary as well. Given this, we can define the energy management problem as follows:

**Definition 3.1.** *The* energy management problem *is a sequential decision making problem where each agent $i$ has to choose appropriate values for its energy budgets at the beginning of each time slot ($t$), so that the amount of sampled, received, and the transmitted information in that round leads the overall system to efficient performance in long-term information collection in dynamic environments.*

Our goal is to efficiently tackle the energy management problem, taking the research requirements into account. Furthermore, the proposed approach should efficiently handle both the trade–off between exploration and exploitation, and the environmental changes as well. In so doing, we distinguish two situations, namely: (i) when energy–harvesting is possible and stable (i.e. the amount of harvested energy does not vary a lot); and (ii) when energy–harvesting is difficult or not possible. In the former case, since energy is repeatedly recharged, the overall energy budget of each agent is not limited. Thus, the agents can comply with the concept of energy–neutrality, in order to indefinitely extend their life span. In contrast, in the latter case, energy–neutrality is not a solution to follow, since the ovelrall energy budget is limited. Instead, the agents have to be energy–aware. Thus, in these situtations, energy management policies must behave differently. Given this, section 3.3.1 discusses the formulated model for the former case, while 3.3.2 formulates the latter case.

### 3.3.1 Energy Management with Energy Harvesting Sensors

In this case, since energy harvesting is possible and stable, we follow the concept of energy–neutrality, in order to guarantee the long–term operation of the network (see chapter 1 for more details). Consequently, the total amount of energy that agent $i$ can use at each slot is equal to the total harvested energy in that slot. Furthermore, since the energy harvesting is stable, we assume that the amount of harvested energy per slot is constant, and we denote it with $B_i$ [2]. Given this, for each time slot $t$ and agent $i$:

$$B_i(t) = B_i \tag{3.6}$$

That is, the amount of energy available for agent $i$ in slot $t$ is $B_i$.

Now, let $B_i^{\mathrm{S}}(t)$, $B_i^{\mathrm{Rx}}(t)$, and $B_i^{\mathrm{Tx}}(t)$ denote the energy budgets that agent $i$ allocates to sampling, receiving and transmitting at time slot $t$, respectively. Given this, for energy consumption of each task, we have the following constraints:

$$\begin{aligned} e_i^{\mathrm{S}} |\mathbf{S}_i(t)| &\leq B_i^{\mathrm{S}}(t) \\ e_i^{\mathrm{Rx}} |\mathbf{Rx}_i(t)| &\leq B_i^{\mathrm{Rx}}(t) \\ e_i^{\mathrm{Tx}} |\mathbf{Tx}_i(t)| &\leq B_i^{\mathrm{Tx}}(t) \end{aligned} \tag{3.7}$$

for each agent $i$. These constraints demonstrate that the energy consumption of each action made by agent $i$ cannot exceed the energy budget of each task given in round $t$. Furthermore, we have:

$$B_i^{\mathrm{S}}(t) + B_i^{\mathrm{Rx}}(t) + B_i^{\mathrm{Tx}}(t) \leq B_i \tag{3.8}$$

In our model, we assume that if a budget allocated to a task is not fully used in slot $t$, then the residual energy is not reusable for the next time slot. That is, for example, if

$$e_i^{\mathrm{Rx}} |\mathbf{Rx}_i(t)| < B_i^{\mathrm{Rx}}(t)$$

then the residual energy (i.e. $B_i^{\mathrm{Rx}}(t) - e_i^{\mathrm{Rx}} |\mathbf{Rx}_i(t)|$) cannot be added to the total energy budget $B_i(t+1)$ of the next slot. This assumption is reasonable, due to the fact that in real world WSNs, the agent allocates energy budget to a task by setting its power consumption level and the time interval in which the task should be performed. That is, the task's energy consumption is rather continuous during that time interval. Given this, even if the energy consumption of of packets successfully sampled, received or transmitted is less than the energy budget, the total allocated budget is used in that time slot.

[2] If the amount of harvested energy varies, then we can take the average, and denote it with $B_i$.

Given all this, our goal is to find a good solution for the energy management problem defined in definition 3.1, by using this model for the case of energy–harvesting sensors. Therefore, in chapter 5, we propose a MAB learning based approach, in order to efficiently tackle this problem (see chapter 5 for more details).

### 3.3.2 Energy Management with Non–Harvesting Sensors

In this case, we assume that the agents cannot harvest energy from the environment. Thus, each agent has to rely on its battery level in the whole operation time. Let $B_i^{\mathrm{T}}(t)$ denote the total residual energy level of the battery of agent $i$ at time slot $t$, we have:

$$B_i^{\mathrm{S}}(t) + B_i^{\mathrm{Rx}}(t) + B_i^{\mathrm{Tx}}(t) \leq B_i^{\mathrm{T}}(t) \tag{3.9}$$

That is, the energy budgets allocated to sampling, receiving, and transmitting at slot $t$ together cannot exceed the total residual energy level. Here, similarly to the case of energy–harvesting sensors, we also assume that the residual energy of each allocated budget is not reusable for the next time slot. Given this, we can update the value of residual energy level of agent $i$'s battery as follows:

$$B_i^{\mathrm{T}}(t+1) = B_i^{\mathrm{T}}(t) - \left(B_i^{\mathrm{S}}(t) + B_i^{\mathrm{Rx}}(t) + B_i^{\mathrm{Tx}}(t)\right) \tag{3.10}$$

Based on this formulation, we also propose a MAB learning based approach in chapter 6, in order to efficiently solve this problem (see chapter 6 for more details).

## 3.4 The Maximal Information Throughput Routing Problem

In this section, we focus on information–centric routing. To this end, we introduce a new routing problem, namely the maximal information throughput routing problem, which is already described in section 1.2. However, for the reader's convenience, we repeat the description of the problem below, in more detail.

Suppose that all the agents have already set their energy budget value for sampling, receiving, and transmitting tasks. In this case, to maximise the value of the total collected information, it is obvious that we need to maximise the total information value of data sampled or relayed by agents that are one hop from the $BS$. The latter, however, is equal to data that is sampled or relayed by agents that are two hops from the $BS$, and so on. That is, beside focusing on efficient energy management, it is also important to maximise the information throughput (i.e. the total transmitted information value)

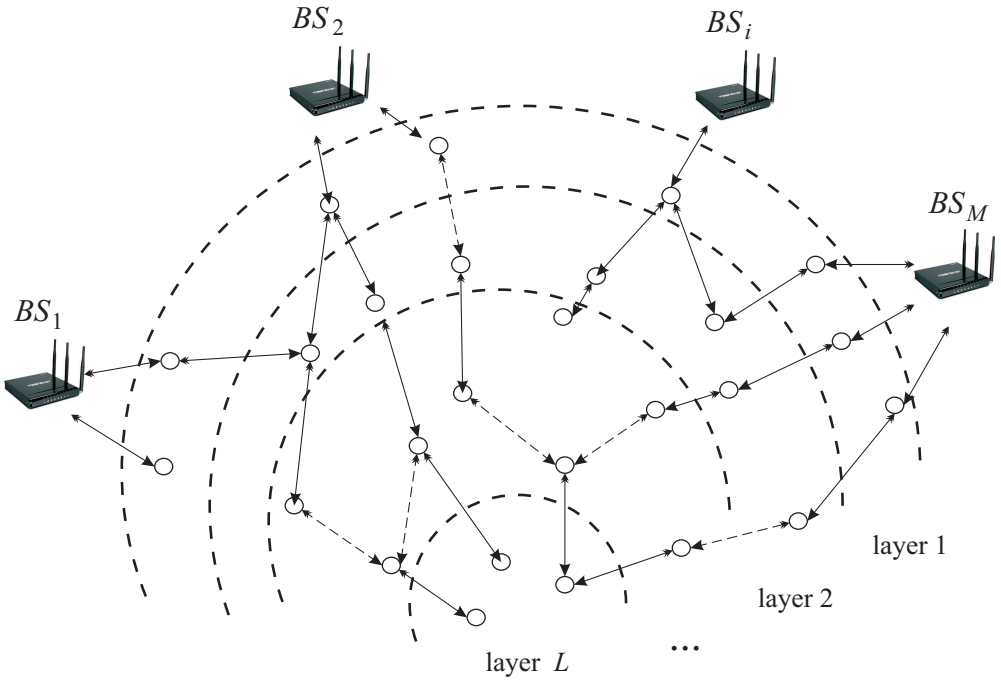



FIGURE 3.1: Network topology in wireless sensor networks.

between neighbouring layers of agents (i.e. group of agents with the same distance from the $BS$) by using efficient routing techniques.

Given this, in our model, we group the nodes into layers such that layer $l$ contains nodes that are $l$ hops from the $BS$. Let $\mathfrak{L}_l$ denote the layer $l$ and $L$ denote the number of layers in the network (see Figure 3.1). Note that the $BS$ itself is layer 0. Now, we can define the routing problem as follows:

**Definition 3.2.** *The* maximal information throughput problem *is the optimisation problem where agents in layer $\mathfrak{L}_l$ together have to perform the maximal total information throughput to layer $\mathfrak{L}_{l-1}$ in round $k$, with respect to the energy budgets on each of the agents.*

In order to solve this problem, we propose an optimal algorithm in chapter 4. Moreover, we will show that the proposed algorithm is efficient in both computational and communication costs.

## 3.5 Summary

In this chapter, we have introduced a generic model for WSN systems. Within this model, we have made several simplifying assumptions related to the radio link models, mobility, data storage and battery recharging. Following this, we have formulated the

adaptive sampling, information content valuation, routing, and energy management policies. For adaptive sampling, and information valuation, we use existing techniques in our model. For routing, we have introduced receiving and transmitting actions with which an agent can achieve high information throughput. For energy management policies, we have defined energy budgets for sampling, receiving and transmitting data. With these budgets, an agent can control its actions adaptively.

In the second part of the chapter, we have formulated the main research objective of our work; that is, to achieve efficient long-term information collection in WSNs. Following this, we have defined two problems, namely the energy management and the maximal information throughput routing problems, that can be separately solved. However, by combining the solutions, we can still achieve efficient long–term information collection. In particular, we have distinguished two cases for the energy management problem, namely: (i) energy management with energy–harvesting sensors; and (ii) energy management with non–harvesting sensors. Then we have described the maximal information throughput routing problem in detail.

Since in both cases of the energy management problem, the underlying routing problem is common, for our convenience, we first study the maximal information throughput routing problem in chapter 4. While the solutions for the cases of energy management problems will be given in chapters 5, and 6, respectively.

# Chapter 4

# Maximal Information Throughput Routing

To achieve efficient long-term information collection in WSNs, we have defined two problems which we aim to solve: namely the (i) energy management problem; and (ii) the maximal information throughput routing problem. Thus, this chapter outlines the work undertaken towards addressing the latter. Specifically, here we describe a decentralised algorithm that allows agents to achieve maximal information throughput between neighbouring layers, with respect to their energy constraints (contribution 1). The algorithm, denoted with MITRA (for maximal information throughput routing algorithm), performs efficient data routing in terms of communication and computational costs.

To this end, in section 4.1, we first formalise the problem. Following this, we describe MITRA in section 4.2 in detail. Then we analyse the performance of MITRA in section 4.3. In particular, we prove that the algorithm optimal in terms of maximising the total information throughput between layers of nodes (see section 4.3.1 for more details). Then we discuss the main advantages and a number of major drawbacks of MITRA (see section 4.3.2). Finally, to demonstrate the algorithm's efficiency in both communication and computational aspects, we analyse simulation results in detail in section 4.4.

## 4.1 Problem Description

Let us use the notations introduced in chapter 3. Recall definition 3.2 in section 3.4; that is, the maximal information throughput routing problem is an optimisation problem in which maximal information throughput at each round has to be achieved between neighbouring layers given constrained sampling, receiving and transmitting capacities of the

agents. Let us note that, according to the model described in chapter 3, these capacities are set by the algorithm that solves the energy management problem (see sections 3.2 and 3.3 for more details). Given this, let $\mathfrak{L}_l$ and $\mathfrak{L}_{(l-1)}$ denote two neighbouring layers, where $1 \leq l \leq L$. For the sake of convenience, hereafter we refer to $\mathfrak{L}_l$ as the sender layer, and to $\mathfrak{L}_{(l-1)}$ as the receiver layer. Similarly, agents in the sender layer will be referred We assume that the current round number is $t$. Thus, we have the following formulation of the maximal information throughput routing problem:

$$\max \left\{ \sum_{i \in \mathfrak{L}_l} \sum_{p \in Tx_i(k)} v\left(p,t\right) \right\} \tag{4.1}$$

That is, we aim to maximise the total information value of transmitted data from layer $l$ to layer $(l-1)$ at round $t$, with respect to the following constraints:

$$E_i^{\mathrm{Tx}} \left| \mathbf{Tx}_i\left(t\right) \right| \leq B_i^{\mathrm{Tx}}\left(t\right) \tag{4.2}$$

for each $i \in \mathfrak{L}_l$. That is, each sender agent cannot exceed its transmitting energy budget during data transmission. Furthermore,

$$E_j^{\mathrm{Rx}} \left| \mathbf{Rx}_j\left(t\right) \right| \leq B_j^{\mathrm{Rx}}\left(t\right) \tag{4.3}$$

for each $j \in \mathfrak{L}_{(l-1)}$, and thus, each receiver agent cannot exceed its receiving budget during data receiving. Finally, constraints described in equations 3.2, 3.3, and 3.5, that express the conservation of information within our setting, have to be taken into account as well.

To find the solution of equation 4.1, we propose a decentralised algorithm, MITRA, described in the following section.

## 4.2 Maximal Information Throughput Routing Algorithm

To find the solution of equation 4.1, we first transform the maximal information throughput routing problem into a graph theory problem, to which we give a centralised optimal solution in section 4.2.1. The reason to first solve the problem in a centralised way is, to understand the mechanism of the algorithm, which which act as the cornerstones for our decentralised algorithm. Following this, based on this solution, we devise a decentralised approach for the problem in section 4.2.2.

### 4.2.1 A Centralised Approach

In order to understand the decentralised algorithm, a graph theory based solution is described in this section. This solution is the cornerstone upon which our decentralised algorithm is based. Note that the centralised version of the maximal information throughput routing problem can also be reduced into a *multiple knapsack problem with assignment restrictions* [Dawande *et al.*, 2000]. However, for the sake of simplicity, here we focus on the graph theory approach.

Given this, let us redefine the routing problem from a graph theory approach as follows. Consider the layers $l$ and $(l-1)$ together as a bipartite directed graph $G$. Each node within these layers is represented by a vertex of $G$. If $i \in \mathfrak{L}_l$ and $j \in \mathfrak{L}_{(l-1)}$ are neighbours, then we connect vertex $i$ with vertex $j$ in $G$ with a directed arc from $i$ to $j$. Each vertex $i$ in the sender layer of $G$ (i.e. the representation of layer $\mathfrak{L}_l$) has a vertex capacity of $c_i = \frac{B_i^{\mathrm{Tx}}(k)}{e_i^{\mathrm{Tx}}}$ and each vertex $j$ in the receiver layer of $G$ (i.e. the representation of layer $\mathfrak{L}_{(l-1)}$) is labeled with the capacity $c_j = \frac{B_j^{\mathrm{Rx}}(k)}{e_i^{\mathrm{Rx}}}$. Furthermore, each vertex $i$ in the sender layer contains the set of transmittable data $\mathbf{Q}_i(k)$. Finally, each data packet $p$ on the sender layer is labeled with a non-negative value $v(p,t)$ (i.e. its information value). The task here is to assign these packets to the arcs with respect to the following constraints:

- Each packet $p \in \mathbf{Q}_i(t)$ can be allocated to at most one of the arcs and this arc must be an outgoing arc from vertex $i$.
- Let $w_{i,j}$ denote the number of packets (i.e. the weight) allocated to arc $(i,j)$. For each vertex $i$ in the sender layer, $\sum_{j \in \mathfrak{L}_{(l-1)}} w_{i,j} \leq c_i$, where $\mathfrak{L}_{(l-1)}$ denotes the set of vertices of the receiver layer. That is, the total weight of the outgoing arcs from $i$ cannot exceed the capacity of vertex $i$. Similarly, for each vertex $j$ in the receiver layer, $\sum_{i \in \mathfrak{L}_l} w_{i,j} \leq c_j$, where $\mathfrak{L}_l$ denotes the set of vertices of the sender layer. That is, the total weight of the incoming arcs to $j$ cannot exceed the capacity of vertex $j$.

Here, our goal is to maximise the total value of the packets allocated to the arcs. In this report, we use a greedy algorithm to achieve the optimal solution for the maximal information throughput routing problem. The reason of choosing such a greedy approach is that greedy algorithms are simple in terms of computational cost, since they only consider the best current option, while they do not look ahead (which step typically increases both memory and computational complexity). To this end, we introduce the following definitions:

**Definition 4.1.** An allocation of packet $p$ is *feasible*, if there is a valid allocation of $p$ to one of the arcs, with respect to the constraints of $G$ mentioned above. For the sake of

simplicity, we refer to this packet as a *feasible packet*. Thus, the *maximal value feasible packet* is a feasible packet with the highest value. Finally, let $\mathfrak{S}$ denote the *current state* of $G$, that is, $\mathfrak{S}$ is the set of current parameter values (i.e. the set of already allocated packets, currently unallocated packets and the current weights of the arcs in $G$).

**Theorem 4.2.** *A greedy method, whereby, for any current state $\mathfrak{S}$ of $G$, if there exists at least one feasible packet, then we allocate one of the maximal value feasible packets (it may be the case that more than one packet has the same value), leads us to the optimal solution of the maximal information throughput routing problem.*

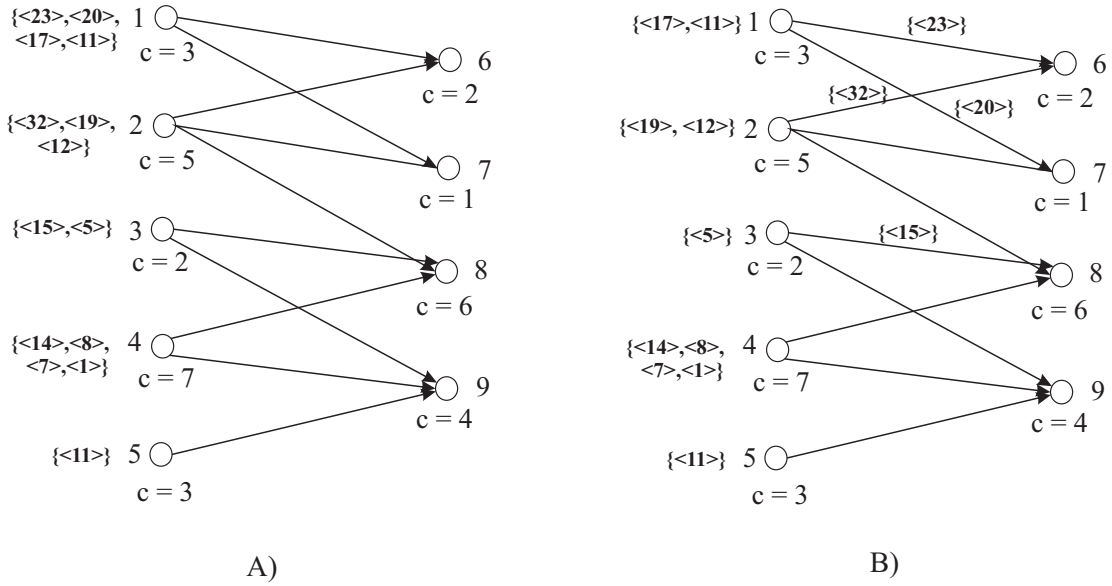



FIGURE 4.1: A) The initial state of $G$. B) The current state of $G$ after the allocation of packets $< 32 >, < 23 >, < 20 >$ and $< 15 >$.

Before we prove theorem 4.2, let us take a look at figure 4.1, that demonstrates the greedy algorithm. In more detail, part A of the figure depicts the initial state of graph $G$. On the sender side, there are 5 vertices, denoted with $1, 2, 3, 4$ and 5. Their capacities are $3, 5, 2, 7$ and 3, respectively. The packets they have are listed in the braces next to the vertices. For instance, vertex 1 contains the list $\{< 23 >, < 20 >, < 17 >, < 11 >\}$. On the receiver side, there are 4 vertices, denoted with $6, 7, 8$ and 9, with capacities $2, 1, 6$ and 4, respectively. Considering the greedy algorithm, in the first step we allocate packet $< 32 >$ of vertex 2, since it is the maximal value feasible packet in the initial state. Here, we allocate it to arc $(2, 6)$. Following that, the next maximal feasible packet is $< 23 >$ of vertex 1 is allocated to arc $(1, 6)$. Then we allocate packet $< 20 >$ of vertex 1 to arc $(1, 7)$. Note that although $< 17 >$ is the next highest value packet, it is not feasible. Thus, in the next step, $< 15 >$ from vertex 3 is allocated to arc $(3, 8)$, and so on. The algorithm stops when no feasible packet is available.

Now, let us turn back to the proof of theorem 4.2. To prove it, here we rely on the following auxiliary lemma:

**Lemma 4.3.** *Given the state $\mathfrak{S}_k$ of graph $G$, the optimal solution $\mathfrak{O}(G, \mathfrak{S}_k)$ of $G$ that can be achieved from state $\mathfrak{S}_k$ (i.e. the maximal result that we can achieve by starting from state $\mathfrak{S}_k$) contains the allocation of one of the maximal value feasible packets of $G$ at state $\mathfrak{S}_k$.*

**Proof of lemma 4.3:** We prove this lemma by using the indirect proof technique. Suppose that there exists such a $\mathfrak{S}_k$ that from this state, $\mathfrak{O}(G, \mathfrak{S}_k)$ does not contain any of the current maximal value feasible packets. Without loss of generality, we can assume that a maximal value feasible packet belongs to vertex 1, namely $p_{\max}$. If in $\mathfrak{O}(G, \mathfrak{S}_k)$ vertex 1 also sends at least one more packet to its neighbours. Then by replacing that one with $p_{\max}$, the total value of allocated packets is increased, which is impossible, since $\mathfrak{O}(G, \mathfrak{S}_k)$ is the optimal solution, thus it should have the highest total value. Therefore, vertex 1 cannot send any packets to its neighbours from state $\mathfrak{S}_k$ in the optimal solution. However, due to the fact that $p_{\max}$ is feasible at state $\mathfrak{S}_k$, there exists such a vertex $j$ that $p_{\max}$ can be allocated to arc $(1, j)$. If in the optimal solution, $j$ accepts data from any of the other senders, then by replacing that packet with $p_{\max}$, the result will also be increased, which is impossible as well, due to the optimality of $\mathfrak{O}(G, \mathfrak{S}_k)$. Thus, there is only one case left, that is, vertex $j$ cannot receive any data in the optimal solution. This implies that the arc $(i, j)$ is still feasible (i.e. packets can be allocated to that arc). Therefore, by allocating $p_{\max}$ to this arc, the optimal value will be increased, which is also a contradiction. Thus, the original assumption, that the optimal solution does not contain any of the maximal value feasible packets at state $\mathfrak{S}_k$, is not true.

□

Thus, now we can prove theorem 4.2 as follows.

**Proof of theorem 4.2:** We start with the initial $\mathfrak{S}_0$ state of $G$, where none of the packets is allocated yet. By using lemma 4.3, the best step we can make here is to choose a maximal value feasible packet and allocate it to an arc. Let $\mathfrak{S}_1$ denote the resulting state. Here, we also have to choose a current maximal value feasible packet greedily, and so on. At each $\mathfrak{S}_k$, according to lemma 4.3, the greedy step always yields the optimal result. We stop when there is no feasible packet available.

□

Given this, lemmas 4.2 and 4.3 yield the following corollaries:

**Corollary 4.4.** *In the optimal solution, if vertex $i$ has $m$ allocated packets, then these packets must be those with the $m$ highest values in $\mathfrak{Q}_i$, which is the set of all packets at vertex $i$.*

**Corollary 4.5.** *In the optimal solution, if node $i$ has $m$ packets allocated to $n$ arcs, then the allocation of these packets can be in any arbirtrary order to those arcs.*

Let a *feasible bundle* denote a bundle of feasible packets from a common vertex, that can be allocated to the same arc at the same time. Due to corollaries 4.4 and 4.5, instead of allocating a single packet at one step as described in the greedy algorithm, we can allocate the bundle of packets that has the highest value among feasible bundles. Moreover, as described in corollary 4.4, this bundle contains the packets with highest values of $\mathfrak{Q}_i$ at vertex $i$. Thus, we can modify the greedy algorithm above so that at each step, the maximal *total value feasible bundle* is chosen to be allocated.

### 4.2.2 The Decentralised Algorithm

The greedy algorithm described in section 4.2.1 is centralised, since to select the maximal value feasible packet or bundle, the algorithm needs global knowledge in order to choose the best feasible bundle. Thus, we cannot use this algorithm in the WSN, due to the need for the decentralised control regime. However, based on the greedy algorithm and corollaries 4.4 and 4.5, we can design a decentralised algorithm that is suitable for our decentralised WSN model. The sketch of this decentralised algorithm looks as follows.

1. Each sender $s_i$ sends the list of its maximal value feasible bundles to its receivers.
2. When receiver $r_j$ has received all the information regarding the possible bundles from its senders, it chooses the best $c(r_j)$ packets among the possible incoming bundles, where $c(r_j)$ is the receiving capacity of $r_j$.
3. Each receiver $r_j$ then sends back the value of the chosen packets to all of its senders.
4. When sender $s_i$ receives all the information regarding the chosen packets of its receivers, it chooses the best feasible bundle that it can send to each of these receivers.
5. Repeat the previous 4 steps until there is no more feasible packet left.

Let us hereafter refer to this algorithm as the maximal information throughput algorithm (MITRA). The pseudo codes for both sender and receiver side of MITRA can be found in algorithms 4.1 and 4.2, respectively. In algorithm 4.1, the pseudo code of the sender side is depicted. Within this algorithm, each sender $s_i$ first checks whether there are

**Algorithm 4.1** MITRA - sender side

```
1: for all s_i do
2:   if ∃ feasible packet then
3:     IsFeasible(s_i) ← TRUE;
4:   else
5:     IsFeasible(s_i) ← FALSE;
6:   end if
7:   ℜ_i ← list of receivers in communication range of s_i;
8:   B_i ← getMaxValueBundle(C(s_i)); {*get C(s_i) highest value packets*}
9:   {notify the receivers}
10:  broadcastMessage(B_i) {*broadcast the value of B_i to all receivers*}
11:  while IsFeasible(s_i) == TRUE do
12:    C(s_i) ← current transmission capacity;
13:    broadcastSENDMessage(); {*broadcast SEND message to all receivers *}
14:    {*listening to meta data messages from receivers, stop listening when all are
       collected or timeout is reached*}
15:    repeat
16:      listenToNotifications();
17:    until all receiver notifications arrive or until timeout;
18:    B_max ← best feasible bundle;
19:    sendBundle(B_max);
20:    Update(C(s_i)); {*update transmission capacity of s_i *}
21:    if C(s_i) ≤ 0 then
22:      {*if no more capacity to send then node is not feasible*}
23:      IsFeasible(s_i) ← FALSE;
24:    end if
25:    if ∄r_j ∈ ℜ_i such that IsFeasible(r_j) == TRUE then
26:      {*if no more capacity on receiver side then node is not feasible*}
27:      IsFeasible(s_i) ← FALSE;
28:    end if
29:  end while
30: end for
```

any feasible packets in its memory (lines $2-6$). First, it broadcasts its highest value feasible bundle to all of the receivers(lines $8-10$). Then, until $s_i$ becomes *infeasible* (i.e. it cannot send any packets), it execute the lines $11-29$, and does the following:

1. It notifies its receivers about its highest value feasible bundle by broadcasting a `SEND` message to all of the receivers(line 13).

2. Then it waits until all the information from its receivers is collected or until a predefined timeout (lines $16-18$). This information states which bundles can be sent to each receiver.

3. Among the possible bundles, it chooses the most valuable one, and sends it to the destination of the bundle (lines $19-20$).

**Algorithm 4.2** MITRA - receiver side

```
1: for all r_j do
2:   C(r_j) ← current receiving capacity;
3:   if C(r_j) > 0 then
4:     IsFeasible(r_j) ← TRUE;
5:   else
6:     IsFeasible(r_j) ← FALSE;
7:     S_j ← list of senders in communication range of r_j;
8:     while IsFeasible(r_j) == TRUE do
9:       {*listening to meta data messages from senders, stop listening when all are
         collected or until timeout *}
10:      repeat
11:        listenToNotifications();
12:      until all sender notifications arrive or until timeout;
13:      L(C(r_j)) ← getMaxValuePackets(C(r_j)); {*collect the highest values from
         senders*}
14:      {notify the senders}
15:      for all sender s_i ∈ S_j do
16:        sendInformationRequestMessage(L(C(r_j)), s_i); {*send REQUEST messages
           to the senders*}
17:      end for
18:      repeat
19:        receiveData(); {*receive data from senders that may arrive this round*}
20:      until all sender data arrive or until timeout;
21:      Update(C(r_j)); {*update receiving capacity of r_j *}
22:      if C(r_j) ≤ 0 then
23:        {*if no more capacity to receive then node is not feasible*}
24:        IsFeasible(r_j) ← FALSE;
25:      end if
26:      if ∄r_j ∈ S_j such that IsFeasible(s_i) == TRUE then
27:        {*if no more capacity on sender side then node is not feasible*}
28:        IsFeasible(r_j) ← FALSE;
29:      end if
30:    end while
31:  end if
32: end for
```

4. Finally, it updates two internal states: namely i) its residual transmitting capacity (line 21), and ii) its feasibility (lines $22-27$). The sender is infeasible when its capacity is 0 or when all of its receivers are infeasible (i.e. the receivers cannot receive any data).

We present the pseudo code of the receivers in algorithm 4.2. Similarly to the sender side, here the receivers first check whether they are feasible (i.e. they can accept more packets) or not (lines $2-6$). Then, until they become infeasible, each receiver $r_j$ does the following:

1. It waits until all of its senders send information about their highest value feasible bundles, or until a predefined timeout (lines $10 - 12$).

2. Then it determines the highest value packets it would accept from its senders (line 13).

3. Following this, it notifies the senders about it decision by broadcasting the list of selected packets' values by sending a `REQUEST` message to each sender from which it wants to receive data packets (lines $15 - 17$).

4. Then it waits until all the required data has arrived or until a timeout (lines $18 - 20$).

5. Finally, it updates two internal states: namely (i) its residual receiving capacity (line 21), and (ii) its feasibility (lines $22 - 29$). The receiver is infeasible when its capacity is 0 or when all of its senders are infeasible (i.e. they cannot send any data).

## 4.3 Performance Analysis

In this section, we focus on the performance analysis of MITRA. Specifically, we prove that the algorithm is optimal, and that its computational and communication costs are not significant. In so doing, we provide a proof of MITRA's optimal behaviour in section 4.3.1. Then we discuss about its major advantages and drawbacks in more detail in section 4.3.2.

### 4.3.1 Optimality of the Algorithm

Apart from its decentralised behaviour, the MITRA algorithm differs from the centralised greedy algorithm described in section 4.2.2 in another important point. That is, in the centralised greedy algorithm, only the maximal value feasible bundle is allocated in each step, while in MITRA, each sender allocates the highest value feasible bundle among its accepted bundles (i.e. bundles which are accepted by at least one of the receivers) concurrently. Thus, it is not obvious that the MITRA is optimal. However, we state the following:

**Theorem 4.6.** *Assuming that the communication between senders and receivers is perfect, that is, none of the messages arrive after the timeout, the MITRA algorithm results in an optimal solution for the maximal information throughput routing problem (i.e. the solution that gives the maximal throughput of information value between the sender and receiver layers).*

**Proof of theorem 4.6:** To prove this theorem, again, we use the indirection technique. Let us assume that the MITRA algorithm given in the previous section is not optimal. That is, the output solution does not maximise the total transmitted information value between the 2 layers. Let $\mathfrak{O}$ denote the output solution of the MITRA algorithm and $\mathfrak{O}_{OPT}$ be one of the optimal solutions. Since we assume that $\mathfrak{O}$ is not optimal, there should be $p_1$ and $p_2$ packets such that in $\mathfrak{O}$ only one of them is allocated in $\mathfrak{O}$ and the other one is allocated in $\mathfrak{O}_{OPT}$. Without loss of generality, we can assume that $p_1$ is allocated in $\mathfrak{O}$ and $p_2$ is allocated in $\mathfrak{O}_{OPT}$. We can also assume that both $p_1$ and $p_2$ are sent to the same receiver $r_j$. It is easy to prove that if $\mathfrak{O} \neq \mathfrak{O}_{OPT}$ then there exist 2 packets such that these assumptions hold.

In particular, there are two cases to investigate. In the first, both $p_1$ and $p_2$ are from the same sender. Due to the fact that the MITRA algorithm is based on the corollary 4.4, it is easy to show that $v(p_1, k) \geq v(p_2, k)$. That is, $p_1$ has a higher information value than $p_2$, since the corollary states that those data which are sent from the sender must be the packets with the highest values in the set of packets of that sender.

In the second case, $p_1$ and $p_2$ are from different senders. Since in MITRA, the receiver uses the greedy approach to allocate possible arriving packets, when $p_1$ is accepted and $p_2$ is not at $r_J$, the only explanation is that $v(p_1, k) \geq v(p_2, k)$.

One can see that in both cases $p_1$ has a higher (or at least the same) value as $p_2$. Thus, if we replace $p_2$ in $\mathfrak{O}_{OPT}$ with $p_1$, we would have a better solution or one that is also optimal. In the first case, we have a contradiction, since $\mathfrak{O}_{OPT}$ is optimal. In the latter case, by replacing all the possible $p_i$-s that are in $\mathfrak{O}$ but not in $\mathfrak{O}_{OPT}$, we would have that $\mathfrak{O}$ is also an optimal solution, which is also a contradiction. Therefore one can see that the original assumption, that is, $\mathfrak{O}$ is not optimal, is not true.

□

### 4.3.2 Advantages and Shortcomings

In this section, we highlight on some of the advantages, and drawbacks of MITRA, respectively. In particular, we study MITRA against the research requirements given in section 1.1. Since we deal with adaptivity by using learning based techniques in energy management, here we only focus on the other requirements, namely: (i) robustness; (ii) computational feasibility; and (iii) limited use of communication. Here, we show that MITRA fulfills the first two requirements. On the other hand, since there is no explicit limit for the number of communication messages MITRA uses during its operation, it may occur that the communication cost is significant. However, by using numerical results of extensive simulations in section 4.4, we will demonstrate that MITRA has low communication cost as well (see section 4.4 for more details).

Given this, we first investigate the robustness and flexibility of MITRA. In MITRA, each single agent can independently learn the topological changes in its surroundings. Specifically, by sending broadcast messages that contains the agent's maximal feasible bundles at each time slot, each agent can detect its active neighbours (by observing who answers to that message), and thus, information regarding topological changes (e.g. link failure or deployment of new agents), can be repeatedly updated. Furthermore, since this broadcast message is sent at every time slot, MITRA always uses the current topology, and does not rely on any previous information from the past. This makes MITRA robust and flexible against topological changes such as link failure, or changes in number of active agents (e.g. node failure or new agents join the network)

In addition, since MITRA uses a simple decentralised policy, its computational cost is not significant. In more detail, in terms of computational complexity, the critical points of MITRA are at line 13 in algorithm 4.2, and at lines 8 and 18 in algorithm 4.1. At the first critical point (i.e. line 13 in algorithm 4.2), each receiver chooses the best packets that it can accept from its senders. This action consists of sorting a simple array and selecting the best $c(r_j)$ values (i.e. $k$-best value selection from an array). Furthermore, at the second critical point (i.e. line 8 in algorithm 4.1), each sender selects its best packets which is also a $k$-best value selection from a simple array. Finally, at the third critical point (i.e. line 18 in algorithm 4.1), to choose the best feasible bundle, each sender has to perform a single best value selection from an array. Assuming that each agent manages its list of packets well (i.e. by sorting these lists frequently), these actions are simple and fast.

On the other hand, in order to achieve maximal throughput of information between layers of agents, the agents have to send a number of coordination messages (the total number of `SEND` and `REQUEST` messages) to each other in MITRA. In particular, each sender repeatedly sends broadcast messages until packet transmission becomes infeasible (i.e. the sender exceeds its transmission capacity limit, or its receivers all become infeasible). Meanwhile, each receiver responses to the aforementioned broadcast messages unless it also becomes infeasible (i.e. the receiver exceeds its receiving capacity limit, or its senders all become infeasible). Furthermore, since the system operates over a finite time interval, each agent can generate a finite number of packets at each time slot. Thus, the total number of packets that can be sent on the sender layer is finite. Given this, it is obvious that MITRA will stop in finite time (i.e. it cannot run without stopping). This indicates that the total number of communication messages is finite as well. However, we cannot guarantee that this number is small; that is, the agent might need a significant amount of coordination messages. Nevertheless, as we will show in section 4.4, the communication cost of MITRA is not significant, compared to the number of agents in the network and to the total size of data to forward.

## 4.4 Empirical Evaluation

In this section, we investigate the communication cost of MITRA, demonstrating that MITRA has a low communication cost. In so doing, we run extensive simulations, using a variety of parameter settings, in order to measure the communication cost of MITRA in different settings. Furthermore, we also compare the performance of MITRA in terms of maximising information throughput, to that of a simple decentralised routing algorithm, in which each agent only uses a single `SEND` and `REQUEST` message for coordination (i.e. this algorithm typically has a lower communication cost than MITRA). Within this comparison, we demonstrate that MITRA significantly outperforms the simple algorithm. Thus, rather using the simple algorithm, it is worthwhile to use MITRA for routing.

To this end, we first set the simulation parameters in section 4.4.1. Following this, we introduce a simple decentralised algorithm, the Naïve Greedy method, as the benchmark algorithm in section 4.4.2, whose performance is then compared to that of MITRA. Finally, the numerical results of these simulations are analysed in section 4.4.3.

### 4.4.1 Parameter Settings

In real world applications, a sensor's typical transmission, receiving and sampling power is 27 mW, 20 mW, and 15 mW, respectively. The solar energy harvesting power is 1800 mW, the maximal data rate is 20 kbps, and the packet size is 10 bytes [Kansal and Srivastava, 2003]. Given this, in our simulations, we use these values to set the parameters of the agents.

Now recall that each receiver, before sending `REQUEST` messages, it chooses the best feasible packets that it can receive from its senders (see line 13 of algorithm 4.2). Thus, in terms of one single round, this can be regarded as a competition in which each sender is competing against the other senders in order to have more of its packets accepted by the receiver in that round. Let us note, however, that there is no real competition between the senders when multiple rounds are considered, since packets that are not accepted in the current round can still be accepted in the next rounds. This implies that the number of communication rounds needed to achieve the maximal result in MITRA is proportional to the average number of accepted packets in each round (i.e. the higher this value is, the fewer rounds are required to be involved). This value, however, depends on the following parameters: (i) the average number of senders per receiver (i.e. the number of competitors); (ii) the receiving capacities on the receiver side; and (iii) the transmitting capacities on the sender side. The first parameter determines the rate of competitiveness, while the latter two determine the total number of acceptable packets, and thus, the average number of packets accepted per round. Therefore, in

order to analyse the performance of MITRA extensively, we vary these parameters in our simulations. Thus, we organise our test cases as follows.

Considering the ratio between the average number of senders and receivers, we have 3 cases, namely: i) the ratio is approximately 1; ii) the ratio is less than 1; and iii) the ratio is greater than 1. Thus, we can divide the test cases into the following groups:

1. *Group 1:* the number of senders $\simeq$ the number of receivers (i.e. the ratio is approximately 1).
2. *Group 2:* the number of senders $\leq$ the number of receivers (i.e. the ratio is less than 1).
3. *Group 3:* the number of senders $\geq$ the number of receivers (i.e. the ratio is greater than 1).

| **Test Cases** | **Group 1:** $N_s \simeq N_r$ | **Group 2:** $N_s \ll N_r$ | **Group 3:** $N_s \gg N_r$ |
|---|---|---|---|
| **Case 1** | $N_s$= 10<br>$N_r$= 10 | $N_s$= 5<br>$N_r$= 10 | $N_s$= 10<br>$N_r$= 5 |
| **Case 2** | $N_s$= 20<br>$N_r$= 20 | $N_s$= 10<br>$N_r$= 20 | $N_s$= 20<br>$N_r$= 10 |
| **Case 3** | $N_s$= 30<br>$N_r$= 30 | $N_s$= 15<br>$N_r$= 30 | $N_s$= 30<br>$N_r$= 15 |
| **Case 4** | $N_s$= 40<br>$N_r$= 40 | $N_s$= 20<br>$N_r$= 40 | $N_s$= 40<br>$N_r$= 20 |
| **Case 5** | $N_s$= 50<br>$N_r$= 50 | $N_s$= 25<br>$N_r$= 50 | $N_s$= 50<br>$N_r$= 25 |
| **Case 6** | $N_s$= 60<br>$N_r$= 60 | $N_s$= 30<br>$N_r$= 60 | $N_s$= 60<br>$N_r$= 30 |
| **Case 7** | $N_s$= 70<br>$N_r$= 70 | $N_s$= 35<br>$N_r$= 70 | $N_s$= 35<br>$N_r$= 70 |
| **Case 8** | $N_s$= 80<br>$N_r$= 80 | $N_s$= 40<br>$N_r$= 80 | $N_s$= 80<br>$N_r$= 40 |
| **Case 9** | $N_s$= 90<br>$N_r$= 90 | $N_s$= 45<br>$N_r$= 90 | $N_s$= 90<br>$N_r$= 45 |
| **Case 10** | $N_s$= 100<br>$N_r$= 100 | $N_s$= 50<br>$N_r$= 100 | $N_s$= 100<br>$N_r$= 50 |

TABLE 4.1: Test cases for performance analysis of MITRA.

In similar vein, in terms of the energy budgets of each agents (which determine the transmitting and receiving capacities of the agents), we can also separate the test cases to the following scenarios: i) the average energy budget on the sender side is higher than

that of on the receiver side; ii) the average energy budget on the sender side is less than that of on the receiver side; and iii) the average energy budget on the sender side is approximately the same as that of on the receiver side. Thus, each test group contains 3 scenarios as follows.

1. *Scenario 1:* The energy budget of the senders is greater than the energy budget of the receivers. That is, $B_{s_i}^{\mathrm{T}} > B_{r_j}^{\mathrm{T}}$. In the simulations, sender energy budgets are random numbers with uniform distribution between 1000 and 1800. Meanwhile, the receiver energy budgets are also set randomly with uniform distribution between 100 and 900.

2. *Scenario 2:* The energy budget of the senders is less than the energy budget of the receivers. That is, $B_{s_i}^{\mathrm{T}} < B_{r_j}^{\mathrm{T}}$. We set the sender energy budgets randomly between 100 and 900 and receiver energy budgets between 1000 and 1800.

3. *Scenario 3:* The energy budget of the senders is similar to the energy budget of the receivers. That is, $B_{s_i}^{\mathrm{T}} \simeq B_{r_j}^{\mathrm{T}}$. In this scenario, both sender and receiver energy budgets are random numbers between 1000 and 1800.

In fact, choosing different numerical values for the range of the energy budgets does not affect much on the communication and computational costs. In particular, changing the value of this range only modifies the size of feasible bundles, but not the number of communication rounds of the MITRA. Given this, we run 10 test cases in each test group, using all of the scenarios. Let $N_{\mathrm{s}}$ and $N_{\mathrm{r}}$ denote the number of senders and receivers, respectively. Table 4.1 shows the test cases that we use for each group.

### 4.4.2 The Benchmark Algorithm

To achieve the optimal result, MITRA needs to use a number of communication messages, which makes MITRA more complex compared to algorithms which use only one pair of `SEND`–`REQUEST` messages, whose performance, however, may not be optimal. Thus, apart from demonstrating that MITRA has low communication cost, it is important to show that: (i) MITRA is not complex compared to simple algorithms with a single communication round; and (ii) the total information throughput delivered from the sender layer to the receiver layer by using MITRA instead of simple algorithms makes MITRA worthwhile. To this end, we choose a benchmark algorithm that is simple (e.g. uses only one pair of `SEND`–`REQUEST` messages), whose performance can be compared to that of MITRA during the simulations. In this report, a simple decentralised algorithm that needs only one communication message on both sender and receiver sides is used as the benchmark method. We refer to it as the *Naïve Greedy* algorithm. This algorithm is naïve since it stops after sending one communication message, and it is greedy because

**Algorithm 4.3** Naïve Greedy Algorithm - sender side

```
1: for all s_i do
2:   if ∃ feasible packet then
3:     ℜ_i ← list of receivers in communication range of s_i;
4:     C(s_i) ← current transmission capacity;
5:     B_i ← getMaxValueBundle(C(s_i)); {*get C(s_i) highest value packets*}
6:     {notify the receivers}
7:     broadcastSendMessage(B_i) {*broadcast the value of B_i to all receivers using
       SEND message*}
8:     {*listening to meta data messages from receivers, stop listening when all are
       collected or timeout is reached*}
9:     repeat
10:      listenToNotifications();
11:    until all receiver notifications arrive or timeout;
12:    B_max ← best feasible bundle;
13:    sendBundle(B_max);
14:    Update(C(s_i)); {*update transmission capacity of s_i*}
15:  end if
16: end for
```

in this algorithm each sender chooses the feasible bundle with highest value. In fact, Naïve Greedy is MITRA limited to one communication message. The pseudo codes of this algorithm for sender and receiver sides can be found in algorithms 4.3 and 4.4, respectively.

In this algorithm, each sender first checks whether it contains a feasible packet (line 2 in algorithm 4.3). If it does, then it does exactly the same as the sender in the MITRA (see algorithm 4.1), but it stop after sending the first communication message.

Similarly, each receiver $r_j$ first checks its receiving capacity (line 3 in algorithm 4.4). If the capacity is higher than 0, then it does exactly the same as the receiver in the MITRA (see algorithm 4.2), but it stop after sending the first communication message.

### 4.4.3 Numerical Results

Using the parameter settings given in section 4.4.1, and the benchmark algorithm described in section 4.4.2, we have the following numerical results: i) results for test group 1 (i.e. when $N_{\mathrm{s}} \simeq N_{\mathrm{r}}$) are depicted in figure 4.2; ii) results for test group 2 (i.e. when $N_{\mathrm{s}} \ll N_{\mathrm{r}}$) are depicted in figure 4.3; and iii) results for test group 3 (i.e. when $N_{\mathrm{s}} \gg N_{\mathrm{r}}$) are depicted in figure 4.4.

Part A of each figure depicts the total communication rounds required by MITRA in the 3 scenarios of each test groups. Furthermore, parts B and C of each figure show the average communication cost (i.e. average number of rounds) of each sender and each

**Algorithm 4.4** Naïve Greedy Algorithm - receiver side

```
1: for all r_j do
2:   C(r_j) ← current receiving capacity;
3:   if C(r_j) > 0 then
4:     S_j ← list of senders in communication range of r_j;
5:     {*listening to meta data messages from senders, stop listening when all are
       collected or timeout is reached*}
6:     repeat
7:       listenToNotifications();
8:     until all sender notifications arrive or timeout;
9:     L(C(r_j)) ← getMaxValuePackets(C(r_j)); {*collect the highest values from
       senders*}
10:    {notify the senders}
11:    for all sender s_i ∈ S_j do
12:      sendInformationRequestMessage(L(C(r_j)), s_i); {*send REQUEST messages to
         the senders*}
13:    end for
14:    repeat
15:      receiveData(); {receive data from senders that may arrive this round}
16:    until all sender data arrive or timeout;
17:    Update(C(r_j)); {*update receiving capacity of r_j*}
18:  end if
19: end for
```

receiver, respectively. Finally, in part D of the figures, we compare the performance of MITRA in terms of information collection with that of Naïve Greedy.

In the first two groups (i.e. when $N_{\mathrm{s}} \simeq N_{\mathrm{r}}$ and $N_{\mathrm{s}} \ll N_{\mathrm{r}}$), the numerical results of the simulations are similar. That is, when the average transmitting capacity on the sender side is higher than the average receiving capacity on the receiver side (i.e. $B_{s_i}^{\mathrm{total}} > B_{r_j}^{\mathrm{total}}$), MITRA produces the worst performance both in terms of the total number of communcation rounds and the average round number of sender and receiver agents (see parts A, B and C of figures 4.2 and 4.3). On the other hand, in the case of $N_{\mathrm{s}} \gg N_{\mathrm{r}}$ (i.e. test group 3), scenario 3 (i.e. $B_{s_i}^{\mathrm{total}} \simeq B_{r_j}^{\mathrm{total}}$) is the worst case scenario in terms of average round number of senders and receivers (see parts B and C of figure 4.4). Meanwhile, MITRA produces the same performance in terms of total number of rounds in this scenario as well as in scenario 2 (i.e. $B_{s_i}^{\mathrm{total}} > B_{r_j}^{\mathrm{total}}$), as depicted in part A of figure 4.4. As we can see in these figures, even though the number of participating agents reaches the number of 100, the total number of rounds that MITRA needs is still below 15. Furthermore, the average round number that a sender needs is less than 3, and the average number that a receiver needs is less than 8. Thus, we can state that the communication cost of MITRA is less than $O(N)$, where $N$ is the number of participating agents in MITRA.

To better understand how good these results are, let us assume that our WSN has 400

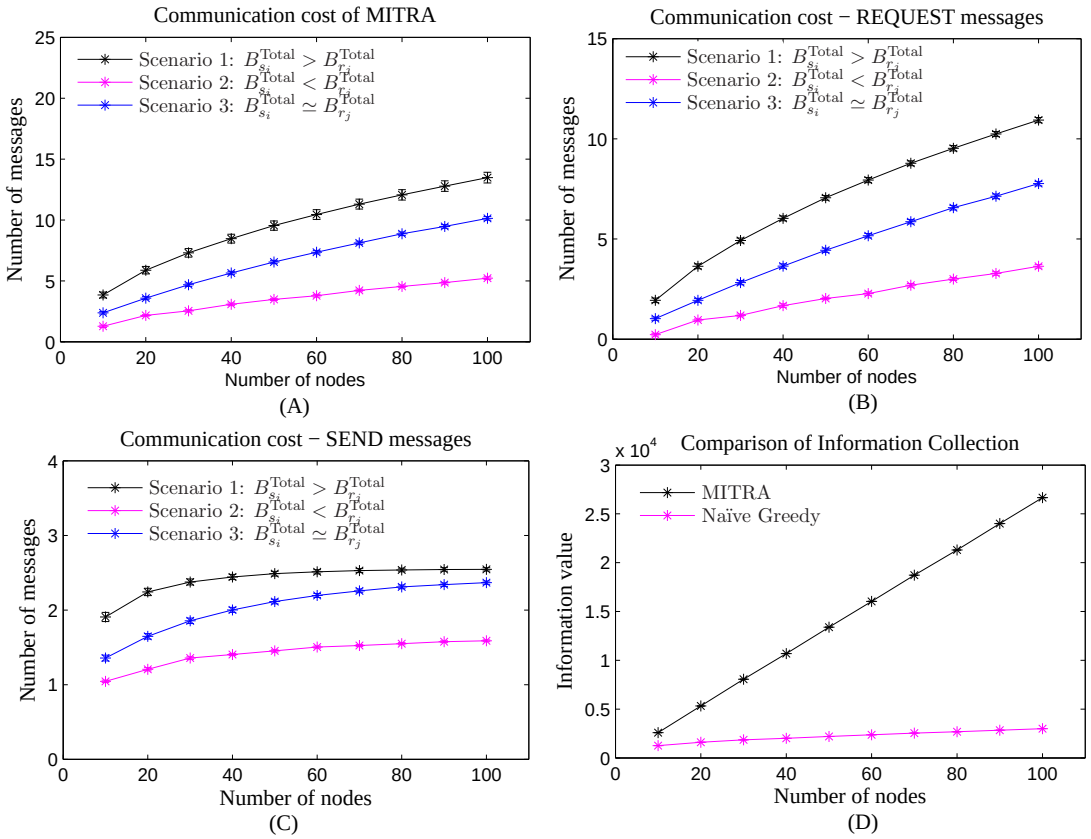



FIGURE 4.2: Numerical results for the case of test group 1 ($N_s \simeq N_r$). (A) The average number of total communication messages per each agent. (B) The average number of `REQUEST` messages per agent. (C) The average number of `SEND` messages per agent. (D) Total information throughput between the layers.

nodes, and the number of layers is 10 (i.e. each node is at most 10 hops far from the $BS$). This implies that on average, each layer contains approximately 40 nodes. Thus, each agent needs at most 10 messages in MITRA to achieve the maximal result. In particular, on average, each agent has to send 3 `SEND` and 7 `REQUEST` messages. Now, suppose that each agent has 10 packets on average to send. Thus, the size of the information value list (see line 8 in algorithm 4.1) to broadcast is 40 bytes (we assume that the information value is stored as a 4-byte real number). In addition, suppose that the size of `SEND` and `REQUEST` messages is 4 bytes each. Then the size of communication messages each agent has to send at each time slot in MITRA is 80 bytes on average. This is small, compared to the total size of real data that the agents typically have to forward in many applications (e.g. in wireless visual sensor networks, the average size of a single data packet is likely to be $10 - 100$ kBytes Kho *et al.* [2009b]).

Let us note that in real world applications, the size of the WSN is typically less than 400 [Cerpa *et al.*, 2001; Mainwaring *et al.*, 2002], thus, MITRA will produce even less communication cost in most applications.

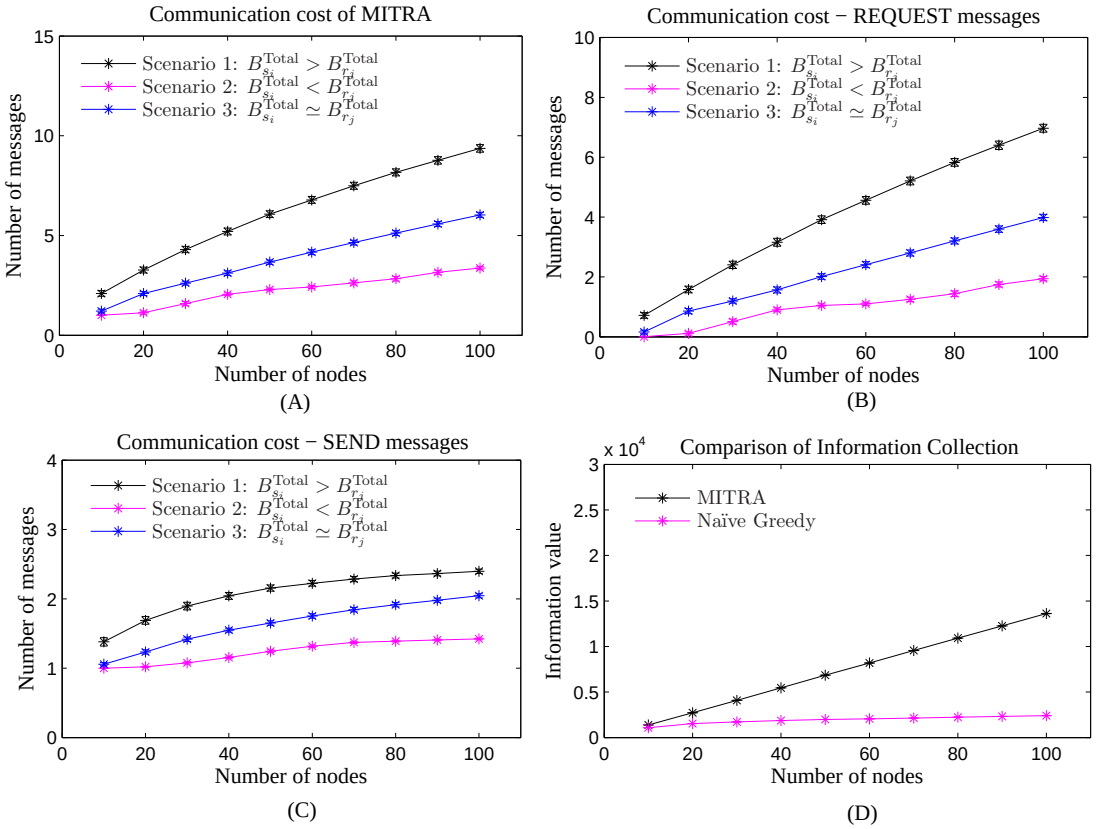



FIGURE 4.3: Numerical results for the case $N_s \ll N_r$. (A) The average number of total communication messages per each agent. (B) The average number of `REQUEST` messages per agent. (C) The average number of `SEND` messages per agent. (D) Total information throughput between the layers.

Finally, to demonstrate that MITRA is worth using instead of simpler algorithms, such as Naïve Greedy, let us consider part D of each figure above. We can see that in each test group, MITRA outperforms Naïve Greedy everywhere. Thus, using the previous example of a WSN with 400 nodes and 10 layers, MITRA in average outperforms Naïve Greedy by 250%.

To conclude the performance analysis of MITRA, we can state that this algorithm incurs a very low communication and computational cost as the number of participating agents increases. Moreover, the average communication cost of each participating agent is not much higher than that of an agent using simple algorithms such as the Naïve Greedy. Finally, the use of MITRA is worthwhile in terms of maximising the information throughput, since it outperforms Naïve Greedy everywhere, typically by a few hundred percent.

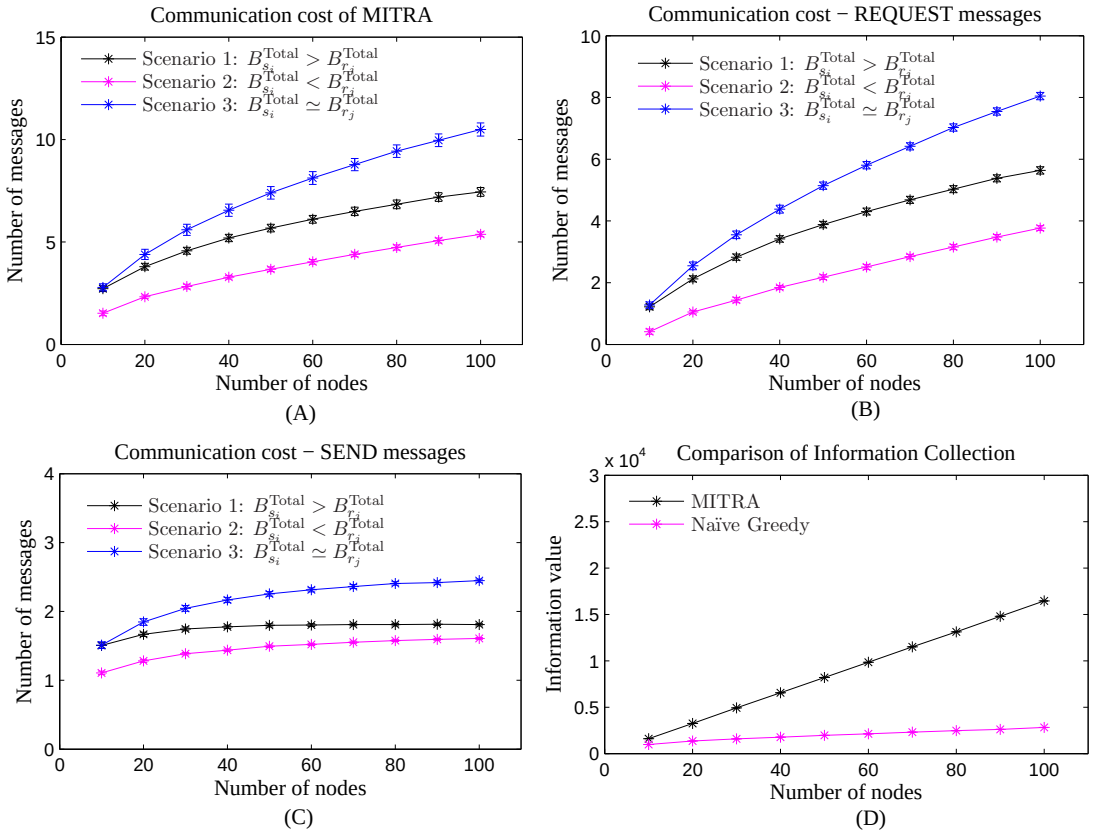



FIGURE 4.4: Numerical results for the case of test group 2 ($N_{\text{s}} \gg N_{\text{r}}$). (A) The average number of total communication messages per each agent. (B) The average number of `REQUEST` messages per agent. (C) The average number of `SEND` messages per agent. (D) Total information throughput between the layers.

## 4.5 Summary

In this chapter, we have described the formalisation of the maximal information throughput routing problem. Following that, we have transformed the maximal information throughput routing problem into a graph theory assignment problem, to which we have given an optimal solution by using lemmas 4.2 and 4.3. We have also constructed a greedy algorithm in order to achieve the optimal result. Then, we introduced two corollaries which act as the cornerstones for our decentralised algorithm. Based on these corollaries, we have constructed MITRA: a decentralised algorithm which optimally solves the maximal information throughput routing problem. This algorithm is the second contribution of this work (see section 1.2 for more details). We have proved the optimality of MITRA, using lemma 4.6, and in order to measure the computational and communication costs of MITRA, we have defined the communication round of each agent participating MITRA. Using this communication round as a metric, we analysed the costs of MITRA via extensive simulations. The numerical results of these simulations demonstrated that MITRA has a low cost both in terms of communication and

computation. Finally, to demonstrate that it is worth using MITRA instead of a simple algorithm that may produce sub-optimal results, we have introduced the Naïve Greedy algorithm as a benchmark approach (existing routing protocols are not suitable for solving the maximal information throughput routing problem). This algorithm, however, has a minimal communication cost, since it only uses one pair of `SEND`–`REQUEST` messages. Thus, we have compared the performance of MITRA in terms of the total information collected compared to that of the Naïve Greedy algorithm. With this comparison, we have shown that MITRA outperforms Naïve Greedy in every testcases.

# Chapter 5

# Energy Management with Energy Harvesting Sensors

In chapter 4, we have introduced an optimal distributed algorithm, called MITRA, which solves the maximal information throughput routing problem, and has low cost both in terms of communication and computation. This algorithm maximises the total information throughput between neighbouring layers given the sampling, receiving and transmitting capacities of each participating agent. However, to maintain the long-term efficiency of MITRA, there is a need to set these capacities adaptively, in order to efficiently handle unpredicted changes of the environment. To this end, in section 3.3, we have defined the energy management problem that covers the aforementioned challenges. In particular, we distinguished two cases for the energy management problem, namely: (i) when the agents can harvest energy from the environment; and (ii) when energy–harvesting is not possible. Given this, this chapter focuses on the energy management problem for the former case. That is, here, the agents can harvest energy from the environment, and thus, they can comply the concept of energy–neutrality, in order to infinitely lengthen their life span. In order to tackle the energy management problem, we propose a novel, MAB based, energy management approach (contribution 2). By using this approach, it is possible to handle the aforementioned challenges efficiently, in order to achieve good long-term performance in information collection of the WSN.

The remainder of this chapter is organised as follows. We first introduce the foundation of the method used for energy management, namely the non–stochastic MAB problem, in section5.1. Specifically, we first describe the model, then we describe a standard state–of–the–art learning algorithm, Exp3, for solving the non–stochastic MAB problem (section 5.1.1). Following this, we discuss the efficiency of Exp3 in section 5.1.2.

In the second part of the chapter, we reduce the energy management problem faced by an agent to a non–stochastic MAB problem, and show that an analogue of the Exp3

algorithm can be used to efficiently solve it (section 5.2). Then we highlight on some of the advantages, and drawbacks of this method in section 5.3, respectively. Finally, in section 5.4, we compare the performances of the proposed MAB learning based approach with state–of–the–art information collection algorithms. Moreover, we also compare the performance of our approach with a centralised optimal non-adaptive algorithm (i.e. its energy budget allocation policy does not change over time), whereby perfect knowledge of the future is assumed to be known.

## 5.1 Non–Stochastic Multi–Armed Bandits

Recall that the non–stochastic MAB model was developed by Auer *et al.* to efficiently handle the situations when the environment is dynamic (see section 2.3.1 for more details). Given this, based on the work of Auer *et al.*, the formulation of this MAB model can be described as follows.

Consider a single agent that operates over a time horizon $T > 0$. Suppose that the agent can perform $K$ different actions, denoted with $1, 2, \ldots, K$. Let $A$ denote the policy of that single agent, which contains a sequence of actions $a_1, a_2, \ldots$, where $a_t$ denotes the action that the agent performs at time step $t$. This policy produces a sequence of rewards $r_{a_1}(1), r_{a_2}(2), \ldots$, where $r_{a_t}(t)$ is the reward provided by action $a_t$ at time step $t$. Without loss of generality, we can assume that $r_{a_t}(t) \in (0, 1]$ for any $t$ time step and $a_t$ action. For any reward assignment and for any $T > 0$, let

$$G_A(T) = \sum_{t=1}^{T} r_{a_t}(t) \tag{5.1}$$

be the *return at time horizon* $T$ of policy $A$ choosing actions $a_1, a_2, \ldots, a_T$. Our goal is to find such policy $A$ that maximises equation 5.1, that is,

$$A_{\max} = \arg\max_{A} G_A(T) \tag{5.2}$$

for given time horizon $T > 0$.

As mentioned in section 2.3.1, the reason that we focus on the deterministic MAB is twofold: (i) the non–stationarity of the rewards; and (ii) we aim to have agent policies that can provide acceptable, but not necessarily optimal results at the beginning of operation. Thus, we need algorithms that are non-stochastic and can provide uniform bounds for any time horizon $T > 0$. To this end, we describe the Exp3 algorithm, which can provide the aforementioned properties. The Exp3 (which stands for "Exponential-weight algorithm for Exploration and Exploitation) was originally proposed in Auer *et al.* [2003]. This algorithm is based on the *Hedge algorithm* presented in Freund and Schapire

[1997], which in turn is a modification of the *weighted majority algorithm* published by Littlestone and Warmuth [1994]. In this section, we investigate a standard version of this algorithm. To this end, let us introduce the following definitions:

**Definition 5.1.** Let the best single action denote the action that provides the highest return at time horizon $T$, that is,

$$G_{\max}(T) = \max_{a_j} \sum_{t=1}^{T} r_{a_j}(t) \tag{5.3}$$

where $j$ is a single action. Given this, the *(weak) regret* of algorithm $A$ is the difference between the performance of $A$ and that of this best single action, defined by

$$G_{\max}(T) - G_A(T). \tag{5.4}$$

In the Exp3 algorithm, actions are randomly chosen using a probability distribution over the set of actions. Thus, to measure the performance of algorithms, in terms of (weak) regret, we aim to determine a policy $A$ that provides a small bound for the following equation:

$$G_{\max}(T) - \mathbb{E}[G_A(T)]. \tag{5.5}$$

Furthermore, this bound have to hold *uniformly* over the time horizon $T$ (i.e., it holds for all $T$ without requiring $T$ as input parameter). Given this, the Exp3 algorithm is described in more details in sections 5.1.1.

### 5.1.1 The Exp3 Algorithm

Before describing the Exp3 algorithm, let us introduce the basic algorithm *Exp3.Sub* that is used as a subroutine in Exp3. This subroutine is shown in algorithm 5.1. At each time slot $t$, Exp3.Sub randomly chooses an action $i$ ($i \leq K$) with probability $p_i(t)$. Then, for all the actions (including the chosen one), it updates the probability $p_i(t)$ for the next slot, proportionally to the current estimate of the expected reward value of the action (i.e. the higher the current estimate is, it chooses that action with a higher probability). In particular, Exp3.Sub maintains a weight value $w_i(t)$ for each action $i$. The update of these weights is shown in lines $6-13$ in algorithm 5.1. Using the new value of the weights, Exp3.Sub adaptively updates each probability $p_i(t)$ as shown in line 3 in algorithm 5.1. This indicates that the higher the current estimate becomes, Exp3.Sub will increase the value of $p_i(t)$, and thus, will choose action $i$ with higher probability, and *vice versa*. Consequently, the algorithm always focuses on the actions with highest current estimates; that is, on those actions which are more likely to be the current best choice. In effect, this update policy guarantees that the agent can efficiently adapt to environmental changes.

**Algorithm 5.1** Algorithm Exp3.Sub

1: **Initialisation:** Let $\lambda \in (0,1]$, and $w_k(1) = 1$ for $k = 1, 2, ..., K$;
2: **for all** $t = 1, 2, \ldots$ **do**
3: Set $p_k(t) = (1-\lambda)\frac{w_k(t)}{\sum_{j=1}^{K} w_j(t)} + \frac{\lambda}{K}$ for $k = 1, 2, \ldots, K$;
4: Draw $a_t$ randomly accordingly to the probabilities $p_1(t), p_2(t), \ldots, p_K(t)$;
5: Receive reward $r_{a_t}(t) \in [0,1]$;
6: **for all** $i = 1, 2, ..., K$ **do**
7: **if** $(i == a_t)$ **then**
8: $\hat{r}_i(t) = \frac{r_i(t)}{p_i(t)}$;
9: **else**
10: $\hat{r}_i(t) = 0$;
11: **end if**
12: $w_i(t+1) = w_i(t) \exp\left(\frac{\lambda \hat{r}_i(t)}{K}\right)$;
13: **end for**
14: **end for**

**Algorithm 5.2** Algorithm Exp3

1: **Initialisation:** Let $t = 1$, and $G_k(1) = 0$ for $k = 1, 2, ..., K$;
2: **for all** $r = 0, 1, 2, \ldots$ **do**
3: $g_r = \frac{(K \ln K) 4^r}{e-1}$;
4: Restart Exp3.Sub choosing $\lambda_r = \min\left\{1, \sqrt{\frac{K \ln K}{(e-1) g_r}}\right\}$
5: **while** $\max_k G_k(t) \leq g_r - \frac{K}{\lambda_r}$ **do**
6: Let $a_t$ be the random action chosen by Exp3.Sub and $r_{a_t}(t)$ the corresponding reward;
7: $G_{a_t}(t+1) = G_{a_t}(t) + r_{a_t}(t)$;
8: $t := t + 1$;
9: **end while**
10: **end for**

The efficiency of Exp3.Sub, however, depends on the value of parameter $\lambda$, since it uses this parameter to calculate the probabilities (see line 3 in algorithm 5.1). Since this $\lambda$ has to be given a priori (line 1 in algorithm 5.1), it may happen that the chosen value is not efficient for the current MAB model. However, we can overcome this shortcoming by adaptively modifying the value of $\lambda$. This modification leads to the Exp3 algorithm, described in algorithm 5.2. In particular, Exp3 divides the time line into rounds. At each round $r$, a new $\lambda_r$ is chosen (see line 4 in algorithm 5.2). At each time step $t$, Exp3 calls subroutine Exp3.Sub (line 6 in algorithm 5.2). Exp3 changes round when the maximal cumulative reward of an arm (i.e. the total amount of rewards Exp3 achieves when that arm is pulled) exceeds a given threshold (line 5 in algorithm 5.2). In this case, it restarts the subroutine Exp3.Sub as well.

### 5.1.2 Regret Bound on the Performance of the Exp3 Algorithm

As mentioned in section 2.3.1, due to the fact that Exp3 has to deal with the dynamic environment, the performance of the algorithm is not as efficient as that of the algorithms proposed for the stochastic case (when the environment is static). That is, they produce worse bounds for the regret, compared to the theoretical optimum of $O(\ln T)$ that can be achieved by statistical model based algorithms. This regret bound is given by the following theorem:

**Theorem 5.2.** *For any $K > 0$,*

$$G_{\max}(T) - \mathbf{E}[G_{\text{Exp3}}] \leq 10.5\sqrt{G_{\max}(T) K \ln K} + 13.8K + 2K \ln K$$

*holds for any assignment of rewards and for any $T > 0$.*

Thus, the expected regret of Exp3 is $O\left(\sqrt{KG_{\max} \ln K}\right)$ uniformly over $T$. The proof of theorem 5.2 can be found in Auer *et al.* [2003].

This regret bound is typically large, since $G_{\max}(T)$ is usually a large number, especially when $T$ is high. However, as we will demonstrate, Exp3 performs well as an energy management algorithm of WSNs in long–term information collection.

## 5.2 Non–Stochastic Multi–Armed Bandits for Energy Management

In this section, we tackle the energy management problem described in section 3.3 by using MAB learning approach. In particular, we focus on the case when agents can harvest energy from the environment (see section 3.3.1 for more details). That is, we use the formal model introduced in section 3.3.1. Recall that within this model, each agent $i$ has an energy budget $B_i$ for each time slot $t$, which is constant over time. Furthermore, agent $i$ has to allocate budgets $B_i^{\text{S}}(t)$, $B_i^{\text{Rx}}(t)$, and $B_i^{\text{Tx}}(t)$ to sampling, receiving and transmitting, respectively. The energy budget allocation, however, has to fulfill equation 3.8.

Given this, we can formulate the energy management problem of a single agent as a non–stochastic MAB as follows. We first define the action set of each agent. Then we determine the reward function of each action. The latter is the mechanism that assigns reward values to the action of the agent at each time slot.

Now, let us consider a decision that agent $i$ can make at round $t$. Since the decision making task for an agent $i$ consists of setting the values of the sampling, receiving and transmitting budgets of that agent at round $t$, we have the following definition:

**Definition 5.3.** Let $n_i^{\mathrm{S}}(t) = \frac{B_i^{\mathrm{S}}(t)}{e_i^{\mathrm{S}}}$, $n_i^{\mathrm{Rx}}(t) = \frac{B_i^{\mathrm{Rx}}(t)}{e_i^{\mathrm{Rx}}}$, and $n_i^{\mathrm{Tx}}(t) = \frac{B_i^{\mathrm{Tx}}(t)}{e_i^{\mathrm{Tx}}}$ denote the sampling, receiving and transmitting capacities (i.e. the maximal number of packets that the agent can sample, receive, or transmit) of agent $i$ at round $t$, respectively. At each action, agent $i$ chooses a combination of the values of those capacities, with respect to the constraint 3.8. Thus, the *3-tuple* $a_i(t) = \left\langle n_i^{\mathrm{S}}(t), n_i^{\mathrm{Rx}}(t), n_i^{\mathrm{Tx}}(t) \right\rangle$ *denotes an action* of agent $i$ at round $t$.

Since the energy consumption limit is fixed for each round $t$, by considering equation 3.8, we can see that the number of options for $a_i(t)$ (i.e. the number of combinations of the capacities) is constant over time as well. Due to the fixed set of actions over time, each agent $i$'s behaviour can be regarded as a MAB model. To this end, we need to define the *action set*. Since the action of agent $i$ is already given in definition 5.3, the set of actions can be easily determined:

$$\mathfrak{A}_i := \left\{ a_i(t) = \left\langle n_i^{\mathrm{S}}(t), n_i^{\mathrm{Rx}}(t), n_i^{\mathrm{Tx}}(t) \right\rangle \right\} \tag{5.6}$$

where

$$n_i^{\mathrm{S}}(t)\, e_i^{\mathrm{S}} + n_i^{\mathrm{Rx}}(t)\, e_i^{\mathrm{Rx}} + n_i^{\mathrm{Tx}}(t)\, e_i^{\mathrm{Tx}} \leq B_i \tag{5.7}$$

That is, $\mathfrak{A}_i$ is the set of 3-tuple of capacities where the total energy consumption does not exceed the energy limit given at each round $k$.

Now, recall that $\mathbf{S}_i(t)$, $\mathbf{Rx}_i(t)$ and $\mathbf{Tx}_i(t)$ are the set of sampled, received and transmitted data packets of agent $i$ at round $t$. Furthermore, $\mathbf{Q}_i(t)$ is the set of total transmittable data packets in the memory (see section 3.2 for more details). Let $\mathbf{Re}_i(t)$ denote the set of residual packets from slot $(t-1)$ (i.e. that are not transmitted until slot $t$) on agent $i$. That is,

$$\mathbf{Re}_i(t) = \mathbf{Q}_i(t) / \mathbf{Tx}_i(t) \tag{5.8}$$

For the reward function, consider the following equation:

$$R_i(t) = \lambda^{d_i - 1} \left\{ \sum_{p \in S_i(t)} v(p,t) - (1-\lambda) \sum_{p \in Re_i(t)} v(p,t) \right\} \tag{5.9}$$

where $d_i$ is the distance of agent $i$ from the $BS$ (in hops), and $\lambda$ is the information discount coefficient. This reward function is evaluated after each action (i.e. energy allocation to sampling, receiving, and transmitting tasks) is chosen. The choice of the sampling capacity determines the information content values agent $i$ can sample at slot $t$, while the choice of receiving and transmitting capacities reflect the total information value of the residual packets. Given this, we state the following:

**Theorem 5.4.** *Using the reward function defined in equation 5.9, the total reward value that the agents in the WSN achieve together over the interval* $[0, T]$ *is equal to the total information content value delivered to the* $BS$ *over that time interval.*

That is, by maximising each agent's total reward over interval $[0, T]$, we can achieve the maximal information collected and delivered to the $BS$.

Now, we prove Theorem 5.4 as follows.

**Proof of theorem 5.4:** For the sake of simplicity, let $\mathfrak{L}_j$ denote the set of agents that are $j$ hops from the $BS$. That is,

$$d_i = j, \forall i \in \mathfrak{L}_j \tag{5.10}$$

Now, consider equation 3.1 in section 3.2. Let us note that since no data can be sampled and forwarded, or received and forwarded at the same round (see section 3.1), no data packets are transmitted or received at round 0 in the whole WSN. Thus, using the notation of chapter 3, the main objective can be rewritten as follows.

$$\max \sum_{t=1}^{T} \left\{ \sum_{p \in Rx_{BS}(t)} v\left(p, t\right) \right\} \tag{5.11}$$

Let us consider a particular member of equation 5.11, which is $\sum_{p \in Rx_{BS}(1)} v\left(p, 1\right)$. This equation determines the total information value that arrives to the $BS$ at round 1. According to our assumptions in section 3.1, no data loss occurs during any transmission. Thus, the amount of received information at the $BS$ is equal to the total amount of information that is transmitted from agents that are 1–hop from the $BS$ at round 1. That is,

$$\sum_{p \in Rx_{BS}(1)} v\left(p, 1\right) = \sum_{j \in \mathfrak{L}_1} \sum_{p \in Tx_j(1)} v\left(p, 1\right) \tag{5.12}$$

Note that the set of transmitted data of $\mathfrak{L}_1$ at round 1 is equal to the set of sampled data at round 0, excluding the set of residual data at round 1 (since there is no received data and the residual set is still empty at round 0). Since newly sampled data does not suffer from information value discounting, the right side of equation 5.12 can be rewritten as the following:

$$\sum_{j \in \mathfrak{L}_1} \sum_{p \in Tx_j(1)} v\left(p, 1\right) = \sum_{i \in \mathfrak{L}_1} \sum_{p \in S_i(0)} v\left(p, 0\right) - \sum_{i \in \mathfrak{L}_1} \sum_{p \in Re_i(1)} v\left(p, 1\right) \tag{5.13}$$

Now, let us consider the second member of equation 5.11, which is $\sum_{p \in Rx_{BS}(2)} v\left(p, 2\right)$. Similarly, this can be rewritten as follows.

$$\sum_{p \in Rx_{BS}(2)} v\left(p, 2\right) = \sum_{j \in \mathfrak{L}_1} \sum_{p \in Tx_j(2)} v\left(p, 2\right) \tag{5.14}$$

However, this is equal to the union of the set of received data, the set of sampled data, and the set of residual data at round 1, excluding the set of residual data of layer 1 at round 2. Furthermore, any of these sets may not be empty. The packets in the sets of received and residual data suffer from value discounting, thus, equation 5.14 is equal to

the following:

$$\begin{aligned}\sum_{p\in Rx_{BS}(2)} v\left(p,2\right) &= \sum_{j\in\mathfrak{L}_1}\sum_{p\in Tx_j(2)} v\left(p,2\right) = \\ &= \sum_{i\in\mathfrak{L}_1}\sum_{p\in S_i(1)} v\left(p,1\right) + \lambda\sum_{i\in\mathfrak{L}_1}\sum_{p\in Re_i(1)} v\left(p,1\right) + \\ &+ \lambda\sum_{i\in\mathfrak{L}_1}\sum_{p\in Rx_i(1)} v\left(p,1\right) - \sum_{i\in\mathfrak{L}_1}\sum_{p\in Re_i(2)} v\left(p,2\right) \end{aligned} \tag{5.15}$$

where $\lambda$ is the discount coefficient of the network. Now let us consider $\sum_{i\in\mathfrak{L}_1}\sum_{p\in Rx_i(1)} v\left(p,1\right)$. Similar to equation 5.12, this can be written as:

$$\lambda\sum_{i\in\mathfrak{L}_1}\sum_{p\in Rx_i(1)} v\left(p,1\right) = \lambda\sum_{i\in\mathfrak{L}_2}\sum_{p\in Tx_i(1)} v\left(p,1\right) \tag{5.16}$$

Using equations 5.15 and 5.16, and replacing $\mathfrak{L}_1$ with $\mathfrak{L}_2$ in equation 5.13, we obtain the following:

$$\begin{aligned}\sum_{p\in Rx_{BS}(2)} v\left(p,2\right) &= \sum_{i\in\mathfrak{L}_1}\sum_{p\in S_i(1)} v\left(p,1\right) - \\ &- \sum_{i\in\mathfrak{L}_1}\sum_{p\in Re_i(2)} v\left(p,2\right) + \lambda\sum_{i\in\mathfrak{L}_1}\sum_{p\in Re_i(1)} v\left(p,1\right) + \\ &+ \lambda\sum_{i\in\mathfrak{L}_2}\sum_{p\in S_i(0)} v\left(p,0\right) - \lambda\sum_{i\in\mathfrak{L}_2}\sum_{p\in Re_i(1)} v\left(p,1\right) \end{aligned} \tag{5.17}$$

In general, if we take the $t^{\text{th}}$ member of equation 5.11, then it can be decomposed as follows. If $t \leq L$, where $L$ is the number of the layers in the network, then:

$$\begin{aligned}\sum_{p\in Rx_{BS}(t)} v\left(p,t\right) &= \sum_{j=1}^{t}\lambda^{j-1}\sum_{i\in\mathfrak{L}_j}\sum_{p\in S_i(t-j)} v\left(p,t-j\right) - \\ &- \sum_{j=1}^{t}\lambda^{j-1}\sum_{i\in\mathfrak{L}_{j+1}}\sum_{p\in Re_i(t-j+1)} v\left(p,t-j+1\right) + \\ &+ \sum_{j=1}^{t}\lambda^{j}\sum_{i\in\mathfrak{L}_j}\sum_{p\in Re_i(t-j)} v\left(p,t-j\right) \end{aligned} \tag{5.18}$$

Let us note that here $\sum_{i\in\mathfrak{L}_j}\sum_{p\in Re_i(0)} v\left(p,0\right) = 0$ for any layer $j$. That is, we can say that the amount of information that arrives to the $BS$ at round $k$ can be decomposed into the sum of data on layer 1 at round $(t-1)$, on layer 2 at round $(t-2)$, and so on. If $t > L$, however, the equation for this case is slightly different, since the decomposition

stops at the last layer of agents. Thus, we have:

$$\begin{aligned}\sum_{p\in Rx_{BS}(k)} v\left(p,k\right) &= \sum_{j=1}^{L}\lambda^{j-1}\sum_{i\in\mathfrak{L}_j}\sum_{p\in S_i(t-j)} v\left(p,t-j\right) - \\ &- \sum_{j=1}^{L}\lambda^{j-1}\sum_{i\in\mathfrak{L}_{j+1}}\sum_{p\in Re_i(t-j+1)} v\left(p,t-j+1\right) + \\ &+ \sum_{j=1}^{L}\lambda^{j}\sum_{i\in\mathfrak{L}_j}\sum_{p\in Re_i(t-j)} v\left(p,t-j\right) \end{aligned} \tag{5.19}$$

Given this, combining equations 5.18 and 5.19, and taking each $t$ into account, we can reformulate our main objective to the following:

$$\begin{aligned}&\sum_{t=1}^{T}\left\{\sum_{p\in Rx_{BS}(t)} v\left(p,t\right)\right\} = \\ &= \sum_{t=1}^{T}\sum_{j=1}^{\min(t,L)}\lambda^{j-1}\sum_{i\in\mathfrak{L}_j}\left\{\sum_{p\in S_i(t-j)} v\left(p,t-j\right) - (1-\lambda)\sum_{p\in Re_i(k-j)} v\left(p,t-j\right)\right\}\end{aligned} \tag{5.20}$$

Consider the core part of equation 5.20 in the braces. Now, using the definition of the reward function in equation 5.9 to replace that part, and recall that the distance of agent $i$ is defined in equation 5.10, we can reformulate 5.20 as follows:

$$\max \sum_{j=1}^{\min(T,L)}\sum_{t=0}^{T-j}\sum_{i\in\mathfrak{L}_j} R_i\left(t\right) \tag{5.21}$$

That is, the original objective can be decomposed to the sum of reward functions of agents on each layer $j$, from round 0 to round $T-j$.

□

Now, using the aforementioned reward function and the action set, the agent's energy management problem can be reduced to a MAB problem. That is, by choosing an action $a_i\left(t\right) = \left\langle n_i^{\mathrm{S}}\left(t\right), n_i^{\mathrm{Rx}}\left(t\right), n_i^{\mathrm{Tx}}\left(t\right)\right\rangle$ at round $t$, agent $i$ sets its capacities. Based on these capacities, this agent receives its reward as described by the reward function. Although the effect of the action on the immediate reward is not explicitly involved in the definition of the reward function given in equation 5.9, the reward value depends on the chosen action, since the current capacities of agent $i$ determine the maximal number of received, transmitted and sampled data of the agent. On the other hand, the reward can be regarded as a random value, since it also depends nondeterministically on the capacities of other neighbouring agents. This is due to the nature of the MITRA algorithm which we use to determine the exact transmitting and receiving data of each agent $i$ at round $k$ (see chapter 4 for more details). Thus, there does not exist an explicit deterministic function that can determine the number of transmitted and received data

packets of each agent, by using the value of current capacities only. Furthermore, since the environment of the WSN is dynamic, the expected information values may change over time, and thus, the reward function has varying expected value as well.

Given this, we can adapt the non-stochastic MAB model to each agent $i$, to address to the energy management problem. That is, we can use the Exp3 algorithm here to determine the best energy allocation to the tasks of the agent. However, we have to guarantee that the reward values are bounded; that is, they can be normalised (i.e. the normalised value is between 0 and 1), which is the key requirements of Exp3. In so doing, we state the following:

**Lemma 5.5.** *Suppose that the information value of each packet is bounded. Thus, the reward function $R_i(t)$ is also bounded with upper and lower bounds.*

**Proof of lemma 5.5:** Let $V$ denote an upper bound on the information values of data packets. That is, for each packet $p$ and round $t$,

$$v(p,t) \leq V$$

Furthermore, consider the capacities of each agent $i$. Due to the energy budget constraints, these capacities are also bounded. That is,

$$\begin{aligned} n_i^{\mathrm{S}}(t) &\leq \frac{B_i^{\mathrm{L}}}{e_i^{\mathrm{S}}} \\ n_i^{\mathrm{Rx}}(t) &\leq \frac{B_i^{\mathrm{L}}}{e_i^{\mathrm{Rx}}} \\ n_i^{\mathrm{Tx}}(t) &\leq \frac{B_i^{\mathrm{L}}}{e_i^{\mathrm{Tx}}} \end{aligned} \tag{5.22}$$

Finally, let us note that $0 \leq \lambda \leq 1$. Now, we first show that the reward function has an upper bound. Since at each round, the number of sampled packets cannot exceed the sampling capacity, we have

$$R_i(t) = \lambda^{j-1} \sum_{p \in S_i(t)} v(p,t) - (1-\lambda) \sum_{p \in Re_i(t)} v(p,t) \leq V n_i^{\mathrm{S}}(t) \leq \frac{V B_i^{\mathrm{L}}}{E_i^{\mathrm{S}}} \tag{5.23}$$

Thus, the reward function has an upper bound. Similarly, we show that it has a lower bound as well. Consider that the number of receiving data packets per round cannot also exceed the receiving capacity, and the amount of transmitted data in a round is at least 0. Given this, at each round, the number of residual packets (i.e. packets which still stay in the memory of the agent) is increasing with at most $\left(n_i^{\mathrm{S}}(t) + n_i^{\mathrm{Rx}}(t)\right)$ at time slot $t$. Thus, the total information value of the residual packets at slot $t$ is bounded as follows.

$$\sum_{p \in Re_i(t)} v(p,t) \leq V \left\{ \sum_{j=0}^{t} \left( n_i^{\mathrm{S}}(i) + n_i^{\mathrm{Rx}}(i) \right) \right\} \leq Vt \left\{ \frac{B_i^{\mathrm{L}}}{e_i^{\mathrm{S}}} + \frac{B_i^{\mathrm{L}}}{e_i^{\mathrm{Rx}}} \right\}$$

Thus, the reward function has the following lower bound:

$$R_i(t) = \lambda^{j-1} \sum_{p \in S_i(t)} v(p,t) - (1-\lambda) \sum_{p \in Re_i(t)} v(p,t) \geq -VT\{\frac{B_i^{\mathrm{L}}}{e_i^{\mathrm{S}}} + \frac{B_i^{\mathrm{L}}}{e_i^{\mathrm{Rx}}}\}$$

where $T$ is the time horizon of the network.

□

Given this, by normalising the reward values, we get the requirement for Exp3; that is, the reward values are within the range $[0, 1]$. Let us hereafter refer to this model as the *multi-armed bandit based energy management* (MAB/EM) model, and with a slight abuse of notation, we refer to the proposed approach (i.e. using Exp3 for allocating energy budgets) as MAB/EM as well.

## 5.3 Performance Analysis

In this section, we focus on the performance analysis of MAB/EM. Specifically, we analyse the regret bound of MAB/EM in the network level (i.e. taking all the agents into account). Furthermore, we highlight the major advantages and disadvantages of MAB/EM. Given this, section 5.3.1 discusses the regret bound of the network's performance. In addition, section 5.3.2 studies the advantages and disadvantages of MAB/EM against the research requirements given in section 1.1.

### 5.3.1 Regret Bounds on the Performance

Recall that the bound of the (weak) regret of Exp3 is $O\left(\sqrt{KG_{\max} \ln K}\right)$ uniformly over $T$, where $K$ is the number of actions a single agent can perform, and $G_{\max}$ is the total reward that the agent can get by choosing the best single action policy (see section 5.1.2 for more details). Thus, each agent $i$ produces this regret compared to the case when it permanently chooses the best possible action. Thus, in the multi–agent environment, the total regret of the system is $O\left(N\sqrt{K^* G^*_{\max} \ln K^*}\right)$ uniformly over $T$, which is the difference between the case when all the agents run Exp3 and the case when each agent permanently chooses its best action. Here, $N$ is the number of agents in the network, $K^*$ is an upper bound of the number of actions at each agent $i$ (different agents may have different numbers of actions), and $G^*_{\max}$ is the universal upper bound of the best action's performance of each agent, over the operation time of the network.

This regret bound, similarly to the case of Exp3's regret bound, is typically significant, compared to the maximal information value the WSN can collect in its operation time.

However, as we will show in section 5.4, by using Exp3 for allocating energy budgets, the agents achieve an efficient performance in long–term information collection in WSNs.

### 5.3.2 Advantages and Shortcomings

From the aspect of computational complexity, MAB/EM is efficient; that is, it has low computational complexity. due to the simplicity of Exp3's update policy.

In addition, by using MAB/EM, the agents do not have to communicate to each other, in order to efficiently coordinate their actions in energy management. In particular, the agents can cooperate with each other by only observing the reward values. For example, if an agent $i$ prefers to receive less data from its neighbours, or send less data (e.g. due to its low energy budget $B_i$, or its current focus on sampling), then it reduces its receiving/transmitting capacity. That is, agent $i$ reduces its information throughput (i.e. the amount of information it forwards towards the $BS$). This action will be observed by the others, when they recognise that sending more data to $i$ produces low reward value in MAB/EM, and thus, they will reduce their data transmission capacity. Similarly, if agent $i$ increases its information throughput, the others will modify their actions to adapt to it as well.

However, this type of cooperation leads to the following potential issue. In particular, by only observing the reward values, the agents may adapt slowly to the changes, since they may have to take an action several times, in order to ascertain that the expected reward value of that action has changed. In such cases, the learning process is slow, and may not determine the optimal behaviour. Given this, our hypothesis is that our approach performs well in less or moderately dynamic environments, when there is enough time to detect the changes that are happening. On the other hand, it may work inefficiently in highly dynamic circumstances, where the changes are transient. In fact, this issue is not only MAB-specific, but it exists for all other learning approaches, since it is hard to detect transient changes. Against this background, in section 5.4, we demonstrate that our hypothesis is correct. Furthermore, our approach still achieves more efficient performance in highly dynamic environments, compared to state-of-the-art non-learning methods.

## 5.4 Empirical Evaluation

Within this section, we study the performance of the MAB/EM approach through extensive simulations. Our main goal is to demonstrate the efficiency of the learning-based approach in long-term information collection in the WSN domain. Therefore, we compare the performance of our approach to a state-of-the-art non-learning method, USAC,

in order to ascertain whether our approach is more efficient. We show that MAB/EM significantly outperforms USAC on average by around 80%.

In addition, we also make a comparison between the performance of MAB/EM with the best fixed policy algorithm, which provides the highest performance among all of the possible fixed policies over a finite horizon. This algorithm, however, is not a feasible solution for our setting, since it is centralised, and it needs complete information knowledge in order to compute the best fixed policy. Moreover, the best fixed policy depends on the value of the finite time horizon, that is, different time horizons may have different best fixed policies. In fact, this best fixed policy algorithm gives a good upper bound for the amount of information collected over a given time horizon. That is, with this comparison, we can measure the (weak) regret of MAB/EM (see section 5.3.1 for more details). We also demonstrate that the performance of MAB/EM is not much worse than that of this best fixed policy algorithm (the difference is around 10%).

To this end, we first set the simulation parameters in section 5.4.1. Following this, we describe the benchmark algorithms in section 5.4.2. Finally, to demonstrate the efficiency of MAB/EM (combined with MITRA), we analyse simulation results in detail in section 5.4.3.

### 5.4.1 Parameter Settings

To compare the performance of the algorithms, we measure the overall amount of information collected by each algorithm over time. To this end, we run each algorithm on several networks with different topologies and environmental characteristics (e.g. the occurrence frequency of the events, or the expected value of information of each event). Then, we take the average of the specific results of the networks. In order to do this, we have to create a number of networks that may differ from each other in both topology and environmental characteristics. Given this, in this section, we describe the parameter settings in order to create these networks and their environments.

In our simulations, the network contains 100 agents, with randomly generated topology. The agents' parameters, such as $e_i^{\mathrm{S}}$, $e_i^{\mathrm{Tx}}$, $e_i^{\mathrm{Rx}}$, and $B_i$, are initially set with similar values to the settings described in section 4.4.1. Furthermore, we set the information discount coefficient $\lambda = 0.9$ [1].

In order to capture the dynamic nature of WSN environments, we set up our simulation environment as follows. We first set three test environments, namely: (i) static (i.e. the environmental characteristics do not change); (ii) moderately dynamic (i.e. the environmental characteristics slowly change); and (iii) extremely dynamic (i.e. the

[1] We also have varied these parameters in other experiments, and we see the same broad patterns.

environmental characteristics rapidly change). For each test case, we divide the simulation time period into intervals called epochs (which are obviously not known to the agents). In each epoch, the environment has different characteristics, which affect the information value of the collected data. Here, we assume that within each epoch, the information value is randomly generated from a normal distribution, but with different mean and variance. For example, in one epoch, agent $i$ can collect data with information value generated from the distribution $\mathcal{N}(5,3)$ (i.e. normal distribution with mean 5 and variance 3). While in the next epoch, the distribution may change to $\mathcal{N}(45,10)$. Given this, we define 5 type of environment characteristics as follows.

1. In this environment, packets are sampled with the information value in the range of 0 and 10, with distribution $\mathcal{N}(5,3)$.
2. Here, packets are sampled with the information value in the range of 10 and 20, with distribution $\mathcal{N}(15,3)$.
3. The information value of each packet is in the range of 20 and 40, with distribution $\mathcal{N}(30,6)$.
4. The information value of each packet is in the range of 30 and 60, with distribution $\mathcal{N}(45,10)$.
5. The information value of each packet is in the range of 60 and 100, with distribution $\mathcal{N}(80,10)$.

In our model, the length of each epoch is 200 time slots for the moderately dynamic case, and 50 for the highly dynamic case. While in the static case, there is only one epoch (i.e. there is no change). When the environment changes its epoch, it randomly chooses one of the aforementioned characteristics types, with the probabilities of 0.5, 0.25, 0.1, 0.1 and 0.05, respectively.

### 5.4.2 Benchmark Algorithms

In this section, we introduce two benchmark algorithms, namely (i) the USAC algorithm; and (ii) a centralised optimal best fixed policy algorithm. In particular, we describe these algorithms in more detail. Furthermore, we also discuss the reason why we choose these algorithms for benchmarking. In so doing, we first start with USAC, which is followed by the description of the best fixed policy algorithm.

In order to demonstrate the efficiency of our proposed approach compared to that of the state–of–the–art, we need to choose a benchmark algorithm that has to fulfill the following requirements:

**Algorithm 5.3** Best Fixed Policy Algorithm

```
1: T ← finite time horizon;
2: for all t = 0, 1, ... T do
3:    run algorithm Simulation with time interval [0, t];
4:    O ← output of Simulation;
5:    for all agent i in the network do
6:       best decision of agent i over time t: m_i ← O_i;
7:    end for
8: end for
```

**Algorithm 5.4** Simulation Algorithm for Best Fixed Policy Algorithm

```
1: Input: time interval [0, t];
2: Output: best decisions array O;
3: for all agent i in the network do
4:    for all m_i' possible decisions of agent i do
5:       for all t' = 0, 1, ... t do
6:          Sample data using the sampling capacity set by decision m_i;
7:          Run both sender and receiver sides of MITRA;
8:       end for
9:    end for
10: end for
11: V_max ← the highest amount of overall information value collected until round t;
12: for all agent i in the network do
13:    O_i ← m_i of agent i with which V_max can be achieved;
14: end for
```

- It must use information content valuation, in order to distinguish important data from unimportant one.
- It must contain an energy management policy, which allocates energy budgets to different sensory tasks of sampling, receiving, and transmitting.

Among state–of–the–art approaches, it has been shown that USAC achieves efficient performance in long term data collection, especially in dynamic environments. Given this, we choose USAC as the first benchmark algorithm, in order to compare the performance of our approach to that of the state–of–the–art. However, since USAC does not follow the concept of energy neutrality, it would be unfair to compare it unmodifiedly, since the agents may deplete during operation (see section 2.1.4 for more details). Therefore, in our simulations, we modify USAC so that each agent $i$, analogously to MAB/EM, cannot exceed the energy budget $B_i$ at each time slot.

The second benchmark algorithm, whereby the best fixed policy over a given finite time horizon is chosen for each agent, is depicted in algorithm 5.3. The algorithm works as follows. First, for each time round $t \in [0, T]$, it runs the auxiliary algorithm Simulation, which returns the best decision of each agent over the time interval $[0, t]$ (line 3 in algorithm 5.3). Here, the best decision of an agent is the one that maximises the overall

amount of information collected until time round $t$. Following this, it sets the best decisions for each of the agents in the network (lines $5-7$ in algorithm 5.3).

The algorithm Simulation (see algorithm 5.4), which is responsible for determining the best decisions of each action given time interval $[0,t]$, works as follows. For each agent $i$ and decision $m_i$ of that agent, it runs the simulation for the network in order to calculate the overall amount of information collected until time round $t$, assuming that all the events until time round $t$ are known (lines $3-10$ in algorithm 5.4). Following this, it chooses the best decision for each agent, that maximises the amount of collected information value (lines $11-14$ in algorithm 5.4).

This algorithm is centralised, since it needs complete knowledge of the environment in order to calculate the overall amount of information collected until time round $t$. Moreover, it also needs to know the future events as well, with which it can compare the performance of each particular in order to choose the best one. Thus, it is not feasible for our setting, since perfect knowledge is impossible, and centralised regime is not allowed here. However, its performance is the optimal (i.e. maximal in information collection) among the fixed policy algorithms. Thus, by comparing its performance with that of the MAB/EM, we can measure the (weak) regret of our proposed algorithm's performance (for more details see section 5.4.3).

### 5.4.3 Numerical Results

In this section, we discuss the numerical results of the simulations, which are run by using the parameter settings given in section 5.4.1, and the benchmark algorithms described in section 5.4.2.

Given this, the total information collected by each algorithm is depicted in figure 5.1. Here, each test case is run with 10000 simulations, each of which has different initial parameter values. The error bars demonstrate the 95% confidence interval. As can be seen, USAC has the best performance at the beginning, due to its efficiently combined sampling and routing behaviour (e.g. it outperforms the others by around 80% in the static case). In particular, USAC increases the sensors' sampling rate in unknown environments or when changes occur, and will decrease this rate as time goes by. However, within USAC, when a packet is chosen to be delivered to the $BS$, it is guaranteed to be delivered to the $BS$, regardless of possible future events. That is, USAC does not consider situations when future events may be more important, and thus, the delivery of that future information has a higher priority. Rather, the resources are already occupied to deliver the current data. This is not the case in our solution, where such delivery guarantees do not hold, and thus, USAC is outperformed by our proposed approach up to 80% in the long run.

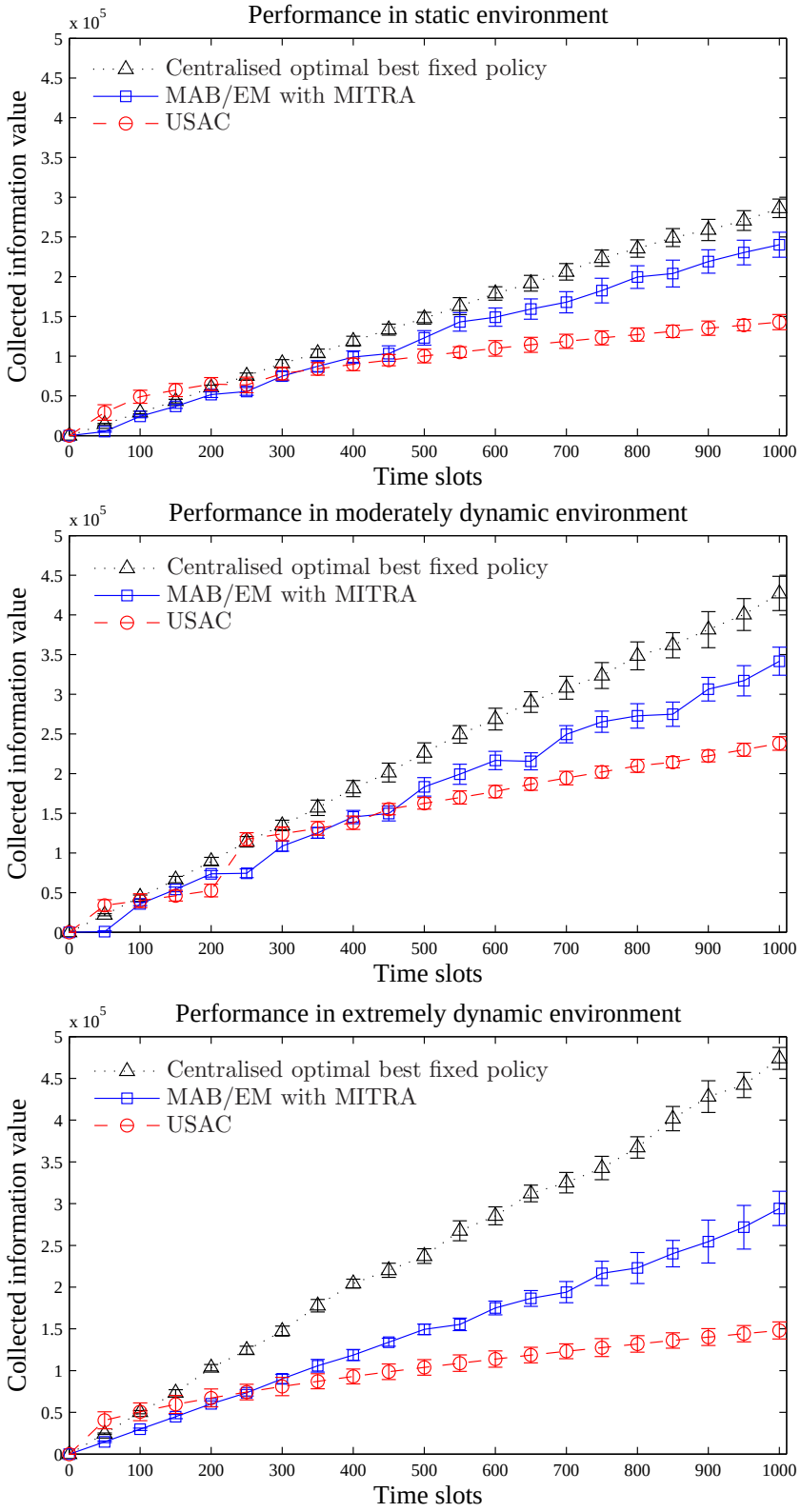



FIGURE 5.1: Information collection in a network with 100 agents, in static, moderately dynamic, and extremely dynamic environments.

Note that, as expected, the efficiency of our approach decreases as the environment becomes more dynamic. Specifically, when the environment is static, the performance of our approach is only slightly worse than that of the optimal policy. However, as the environment becomes more dynamic, the performance of our approach degrades, compared to that of the optimal. The reason is that whenever changes occur, MAB/EM has to find a new, efficient setting again. That is, it has to learn how to adapt to the new environment characteristics. This learning period decreases the performance of the algorithm, since bad settings have to be explored before good ones can be exploited. Nevertheless, in all cases, our algorithm outperforms USAC; that is, it is still able adapt more efficiently to the environmental changes.

To conclude the performance evaluation of MAB/EM, we can state that it significantly outperforms USAC, and it approaches the performance of the best fixed policy algorithm with an efficient approximation rate. Since the latter is just an idealistic algorithm, which is infeasible for our setting, it is reasonable to use learning–based techniques in order to achieve efficient long-term information collection in the WSNs.

## 5.5 Summary

In this chapter, we have proposed a MAB based energy management approach for the case of energy–harvesting sensor agents. In so doing, we first have discussed the non–stochastic MAB problem, which was originally developed by Auer *et al.* [2003]. In more detail, we have discussed this MAB model, then we have described the Exp3 algorithm. Following this, we have studied the performance regret of Exp3.

We then have described our energy management problem as a non–stochastic multi–armed bandit (MAB) problem (Contribution 2). In particular, we have defined the action set of each agent, whereby each action is a combination of energy budget allocation to the tasks of the agent. Furthermore, we have also determined the reward function. We have proved that, by maximising the reward function, the agents together maximise the total information collection at the $BS$ as well. Then we have described the application of the Exp3 algorithm for the MAB/EM problem. Following this, we have discuss the performance of this approach, by analysing its performance regret, and highlighting its advantages and drawbacks, respectively.

Finally, to demonstrate the efficiency of our approach, we have run a number of simulations. In more detail, we first have defined the parameter settings for both test network topology and test environment. Then we have introduced the benchmark algorithms, namely: (i) the USAC algorithm; and (ii) a centralised best fixed policy algorithm. Following this, we have compared the performance of the MAB/EM in terms of the total amount of the collected information value over time horizon $T = 1000$ to that of the

benchmark algorithms. With this comparison, we have shown that MAB/EM significantly outperforms the USAC algorithm on average and efficiently approaches the best fixed policy algorithm. That is, we have shown that by using learning–based techniques, we can achieve more efficient performance of long–term information collection in the WSNs, which is the primary objective of this report. Furthermore, we also have shown that by using the learning approach, our proposed algorithm can perform well in any kind of environments, without using any prior knowledge of that environment.

# Chapter 6

# Energy Management with Non–Harvesting Sensors

In chapter 5, we have introduced a MAB learning based approach for efficient energy management in WSNs, in order to achieve good performance in long–term information collection. However, that approach exploits the fact that the agents can harvest energy from the environment. In particular, due to their capability of energy–harvesting, they can comply with the concept of energy–neutrality, in order to indefinitely extend their life span, which is especially important when information collection has to operate over a prolonged period of time. Furthermore, the amount of harvested energy over a single time slot can be regarded as constant. Given this, within the energy management problem, the action set of each agent is fixed over time (see section 5.2 for more details). Thus, we can formulate the energy management problem that a single agent faces as an existing MAB model, the non–stochastic MAB, in order to achieve efficient long–term information collection.

However, in many real world applications, it may occur that energy–harvesting is difficult or not possible (e.g. the energy recharging module is malfunctioning, or no energy sources are provided). In this case, the agents cannot follow the concept of energy–neutrality, and thus, they cannot use the aforementioned MAB model to manage the energy budget allocation to the sensory tasks (see section 3.3 for more details). Against this background, this chapter focuses on the energy management problem for non–harvesting sensor agents. In so doing, we first introduce a new MAB model, the budgeted MAB with pulling cost, which forms the foundation of our MAB based approach for the energy management problem (contribution 3). We then devise the budgeted $\epsilon$–first approach, in order to efficiently deal with the budgeted MAB problem with pulling cost (contribution 4). Following this, we describe our MAB based energy management approach, based on the aforementioned new MAB model. By using this approach, it is

possible to achieve good long-term performance in information collection when energy–harvesting is not possible.

Against this background, the remainder of this chapter is organised as follows. In section 6.1, we introduce the problem budgeted MAB with pulling cost. Specifically, we first formulate the model, then we introduce a simple, but efficient approach, the $\epsilon$–first policy for that MAB problem. Then we analyse both theoretically and numerically the performance of the the $\epsilon$–first policy. Following this, we discuss the MAB based energy management approach in section 6.2. Finally, we analyse the performance of this approach in section 6.3.

## 6.1 Budgeted Multi–Armed Bandits with Pulling Cost

As mentioned in section 2.3, the multi–armed bandit is a classical problem in decision theory, and presents one of the clearest examples of the trade–off between exploration and exploitation in reinforcement learning. However, this standard MAB gives an incomplete description of the sequential decision–making problem facing an agent in many real–world scenarios. To this end, a variety of other related models have been studied recently, and, in particular, a number of researchers have focused on MABs with a limited exploration budget. Specifically, they consider cases where pulling an arm incurs a pulling cost, whose currency is not commensurable with that of their rewards. The agent's exploration budget then limits the number of times it can sample the arms, thus defining an initial exploration phase, during which the agent's sole goal is to determine the optimal arm to pull during the subsequent cost–free exploitation phase (see section 2.3.2 for more details).

Building on these works, in this section, we consider a further limitation on the agent's behaviour, in which pulling an arm is costly, and *both* the exploration and exploitation phases are limited by a budget. This type of limitation is well motivated by real world applications. For example, consider our WSN application, whereby a sensor node's energy allocation to sensory tasks, such as sampling or data forwarding, can be regarded as an arm of a MAB model (see chapter 5 for more details). Now, because the battery charge is limited, both the exploration (i.e. learning the rewards for different arms) and exploitation phases (i.e. taking the optimal arms given reward estimates) are budget limited as well. In particular, as mentioned earlier, this chapter focuses on the problem of energy management with non–harvesting sensors (see section 6.2 for more details).

Another related example of a this type of problem is that of a company that aims to advertise itself on the web. In so doing, it has a limited budget for renting on–line advertising banners on any of a number of web sites, each of which charges a different rental price. The company wishes to maximise the number visitors who click

on its banners, however, it does not know the click–through rate for banners on each site. As such the company needs to estimate the click–through rate for each banner (exploration), and then to choose the combination of banners that maximises the sum of clicks (exploitation). In terms of the model described above, the price of renting an advertising banner from a website is the pulling cost of an arm, and the click–through rate of a banner on a particular website is the true reward for pulling that arm, which is unknown at the outset of the problem.

Previous MAB models, however, are not suitable in either of these example problems, because they do not consider the case where the agent's exploration and exploitation activities are both limited by a budget. However, as the examples above show, this type of problem is commonly encountered in real world applications of reinforcement learning agents. In order to address this gap, in this chapter we introduce a new version of MAB, a *budgeted MAB with pulling cost* (i.e. pulling an arm is costly, and both exploration and exploitation phases are budget–limited). The overall budget limit differentiates it from previous models in that the overall number of arms pulled is *finite*. Consequently, the optimal solution is not to repeatedly pull the optimal arm *ad infinitum*, but to pull the combination of arms that maximises the reward and fully exploits the budget. To see this, first suppose the expected rewards for pulling the arms are known. In this case, a MAB with an overall budget limit reduces to an unbounded knapsack problem [Andonov *et al.*, 2000], as follows. Pulling an arm corresponds to placing an item into the knapsack, with the arm's expected reward equal to the item's value and the pulling cost the item's weight. The overall budget is then the weight capacity of the knapsack. Given this, the optimal combination of items for the knapsack problem is also the optimal combination of pulls for our MAB problem. This difference in desired optimal solution from existing MAB problems means that (now faced with unknown rewards), when defining a decision–making policy for our problem, we must be cognizant of the fact that an optimal policy will involve pulling a combination of arms. As such, it is not sufficient to learn the expected reward of only the highest–value arm; we must also learn the other arms' rewards, because they may appear in the optimal combination. Importantly, we cannot simply import existing policies that are based on upper confidence bound or $\epsilon_n$–greedy arguments, because they concentrate on learning only the value of the highest expected reward arm, and so will not work in this setting.

For example, consider a three–armed bandit, with arms $X$, $Y$ and $Z$ that have true expected reward–pulling cost value pairs {9, 4}, {1.5, 1} and {2, 1}. Suppose the remaining budget is 15. At this point, the optimal solution is to pull arms $X$ and $Z$ three times each, giving an expected total reward of 33. Now consider the case where the estimates of expected reward for arms $Y$ and $Z$ have been generated using a less efficient exploration policy, and as such are not particularly accurate (whereas the estimate of $X$ is considerably more refined). Specifically, imagine a situation where the estimates are

1.76 and 1.74 for arms $Y$ and $Z$, respectively. In this case, the proposed best solution would be to pull arms $X$ and $Y$ three times each, which on average returns a suboptimal payment with real expected total reward of 31.5. It is clear from this example that better estimates of the mean reward of all arms would allow the agent to determine the optimal combination of pulls. It is also evident that new techniques must be developed for this new problem, which do consider the combinatorial aspect of the optimal solution to the MAB problem investigated in this paper.

Against this background, we propose a *budgeted $\epsilon$–first approach* for our MAB, in which the first $\epsilon$ of the overall budget $B$ is dedicated to exploration, and the remaining portion is dedicated to exploitation. In more detail, we split the budget into two parts, $\epsilon B$ reserved exclusively for exploration, and $(1-\epsilon)B$ for exploitation. In the exploration phase, the agent estimates the expected rewards by *uniformly* sampling the arms (i.e. the arms are sequentially pulled one after the other). Then, in the exploitation phase, it calculates the optimal combination of pulls based on the estimates generated during the exploration phase. The key benefit of this approach is that we can easily measure the accuracy of the estimates associated with a particular value of $\epsilon$, because all of the arms are sampled the same number of times. Hence, we can control the expected performance regret (i.e. the difference between the optimal and the proposed combination of pulls) as a function of $\epsilon$, which gives us a method of choosing an optimal $\epsilon$ for a given scenario. Furthermore, it also gives us a bound on the performance of the $\epsilon$-first approach to the MAB problem with an overall budget.

To this end, we first introduce the model of the budgeted MAB with pulling cost in section 6.1.1. Following this, we discuss the budgeted $\epsilon$-first policy in section 6.1.2. Section 6.1.3 studies the theoretical regret bounds of performance of the budgeted $\epsilon$-first algorithm. Finally, we evaluate the performance of the budgeted $\epsilon$-first policy through extensive simulations in section 6.1.4.

### 6.1.1 Model Description

In this section, we describe the budgeted MAB with pulling cost. In more detail, we consider a $K$–armed bandit problem, in which only one arm can be pulled at any time by the agent. Upon pulling arm $a_t$ of the machine at time slot $t$, the agent pays a pulling cost, denoted $c_{a_t}$, and receives a reward drawn from a distribution associated with that specific arm. The pulling cost $c_{a_t}$ depends only on the arm $a_t$, and is constant over time. Note that in our model, the only restriction on the distribution of these rewards is that they have bounded supports. This assumption is reasonable, since in real–world applications, reward values are typically bounded. Given this, the agent's goal is to maximise the sum of rewards it earns from pulling the arms of the machine. Let $\mu_a$

denote the mean value of the reward that we get by pulling arm $a$; that is:

$$\mu_a = \mathbb{E}\left[r_a\right] \tag{6.1}$$

where $r_a$ is the reward value we can get by pulling arm $a$. However, initially, the agent has no knowledge about $\mu_a$ of each arm $a$, so must learn these values in order to deduce a policy that maximises its sum of rewards. Furthermore, in our model, the agent has a cost budget $B$, which it cannot exceed during its operation time. That is, the total cost of pulling arms cannot exceed this budget limit. Given this, the agent's objective is to find the optimal sequence of pulls, that maximises the expectation of the total reward the agent can achieve, without exceeding the cost budget $B$.

Formally, let $A = \{a_1, a_2, a_3 \dots\}$ be a *finite* sequence of pulls, where $a_t$ denotes the arm pulled at time $t$. The set of possible sequences is limited by the budget. That is, the total cost of the sequence $A$ cannot exceed budget $B$:

$$\sum_{a_t \in A} c_{a_t} \leq B$$

Let $R\left(A\right)$ denote the total reward the agent receives by pulling the sequence $A$. The expectation of $R\left(A\right)$ is:

$$\mathbb{E}\left[R\left(A\right)\right] = \sum_{a_t \in A} \mu_{a_t}$$

Then, let $A^*$ denote the optimal sequence that maximises the expected total reward, which can be formulated as follows:

$$A^* = \arg\max_A \mathbb{E}\left[R\left(A\right)\right] = \arg\max_A \sum_{a_t \in A} \mu_{a_t} \tag{6.2}$$

Note that in order to determine $A^*$, we have to know the value of $\mu_{a_t}$ in advance, which does not hold in our case. Thus, $A^*$ represents a theoretical optimum value, which is unachievable in practice.

However, for any sequence $A$, we can define the regret for $A$ as the difference between the expected cumulative reward for $A$ and the theoretical optimum $A^*$. More precisely, letting $L\left(A\right)$ denote the regret function, we have:

$$L\left(A\right) = \mathbb{E}\left[R\left(A^*\right)\right] - \mathbb{E}\left[R\left(A\right)\right] \tag{6.3}$$

Given this, our objective is to derive a method of generating a sequence, $A$, that minimises this regret function for the class of MAB problems defined above.

### 6.1.2 The Budgeted $\epsilon$–First Approach

In this section we introduce our $\epsilon$–first approach for the MAB with budget limits problem. The structure of the $\epsilon$–first approach is such that the exploration and exploitation phases are independent of each other, so in what follows, the methods used in each phase are discussed separately. Now, several combinations of methods can be used for both exploration and exploitation. However, we investigate only one exploration and one exploitation method in detail, while the study of other methods is left for future work. In particular, for the former, we use a *uniform pull sampling method*, in which all of the arms are uniformly pulled until the exploration budget is exceeded, because we can put a higher bound for the performance regret of an $\epsilon$–first algorithm using this exploration method than for other methods. In the exploitation phase, due to the similarities of our MAB to unbounded knapsack problems when the rewards are known, we use the *reward–cost ratio ordered greedy* method, a variant of an efficient greedy approximation algorithm for knapsack problems.

#### 6.1.2.1 Uniform Pull Exploration

In this phase, we uniformly pull the arms, with respect to the exploration budget $\epsilon B$. That is, we sequentially pull all of the arms, one after the other, until the budget is exceeded. For the sake of simplicity, hereafter we refer to the $i^{\text{th}}$ arm as $i$ ($1 \leq i \leq K$). Letting $n_i$ denote the number of pulls of arm $i$, we have:

$$\left\lfloor \frac{\epsilon B}{\sum_{j=1}^{K} c_j} \right\rfloor \leq n_i \leq \left\lfloor \frac{\epsilon B}{\sum_{j=1}^{K} c_j} \right\rfloor + 1 \tag{6.4}$$

The reason of choosing this method is that, in order to bound the regret of the algorithm, since we do not know which arms will be pulled in the exploitation phase, we need to treat the arms equally in the exploration phase. Hereafter we refer to the sequence sampled by the uniform algorithm as $A_{\text{uni}}$.

#### 6.1.2.2 Reward–Cost Ratio Ordered Exploitation

We first introduce the foundation of the method used in this phase of our algorithm, the unbounded knapsack problem. We then describe an efficient approximation method for solving this knapsack problem, which we subsequently use in the second phase of our budgeted $\epsilon$-first algorithm.

The unbounded knapsack problem is formulated as follows. Given $K$ types of items, each item type $i$ has a corresponding value $v_i$, and weight $w_i$. In addition, there is also a knapsack with weight capacity $B$. The unbounded knapsack problem selects integer

units of those items that maximise the total value of items in the knapsack, such that the total weight of the items does not exceed the knapsack weight capacity. That is, the goal is to find the non–negative integers $x_1, x_2, \ldots, x_K$ that

$$\max \sum_{i=1}^{K} x_i v_i \quad \text{s.t.} \quad \sum_{i=1}^{K} x_i w_i \leq C$$

This problem is *NP*–hard, however, near–optimal approximation methods have been proposed to solve this problem (a detailed survey of these algorithms can be found in [Andonov *et al.*, 2000]).

In particular, here we make use of a simple, but efficient approximation method, the *density ordered greedy* algorithm, which has $O(K \log K)$ computational complexity, where $K$ is the number of type of items [Kohli *et al.*, 2004]. This algorithm works as follows: Let $v_i/w_i$ denote the *density* of item type $i$. At the beginning, we sort the item types in order of the value of their density. This needs $O(K \log K)$ computational complexity. Following this, in the first round of this algorithm, we choose the item type with the highest density and select as many units of this item as are feasible, without exceeding the knapsack capacity. Then, in the second round, we identify the item type with the highest density among the remaining feasible items (i.e. items that still fit into the residual capacity of the knapsack), and again select as many units as are feasible. We repeat this step in each subsequent round, until there is no feasible item left. Clearly, the maximal number of rounds is $K$.

We now reduce the problem faced by an agent in the exploitation phase of our algorithm to the unbounded knapsack problem, and show that an analogue of the density ordered greedy algorithm can be used to efficiently solve it. To begin, recall that in the exploitation phase, the agent makes use of the expected reward estimates from the exploration phase. Let $\hat{\mu}_i$ denote the estimate of $\mu_i$ after the first phase. Given this we aim to solve the following integer program:

$$\max \sum_{i=1}^{K} x_i \hat{\mu}_i \quad \text{s.t.} \quad \sum_{i=1}^{K} x_i c_i \leq (1-\epsilon) B$$

where $x_i$ is the number of pulls of arm $i$ in the exploitation phase. In this case, the ratio of an arm's reward estimate to its pulling cost, $\hat{\mu}_i/c_i$, is analogous to the "density" of an item, because it represents the reward for consuming one unit of the budget, or one unit of the carrying capacity of the knapsack. As such, the problem is equivalent to the knapsack problem above, and in order to solve it, we can use a *reward–cost ratio* (density) ordered greedy algorithm. Hereafter we refer to the sequence pulled by this algorithm as $A_{\text{greedy}}$.

Now, let $x_i^{\text{greedy}}$ denote the solution for arm $i$ given by the value density ordered algorithm. The expected total reward is:

$$\mathbb{E}\left[R\left(A_{\epsilon-\text{first}}\right)\right] = \sum_{i=1}^{K}\left(n_i + x_i^{\text{greedy}}\right)\mu_i$$

where $A_{\epsilon-\text{first}} = A_{\text{uni}} + A_{\text{greedy}}$ denotes the sequence of pulls resulted by our proposed algorithm.

### 6.1.3 Regret Bounds on the Performance of the $\epsilon$–First Policies

In this section, we first derive an upper bound for the performance regret of *any* exploration policy and the reward–cost ratio ordered greedy exploitation algorithm (i.e. the upper bound is independent of the choice of the exploration algorithm). We then refine this bound for the specific case of uniform pull exploration, in order to have a more practical and efficient bound.

Recall that both $A_{\text{uni}}$ and $A_{\text{greedy}}$ together form sequence $A_{\epsilon-\text{first}}$, which is the policy generated by our $\epsilon$-first algorithm. The expected reward for this policy is then:

$$\mathbb{E}\left[R\left(A_{\epsilon-\text{first}}\right)\right] = \mathbb{E}\left[R\left(A_{\text{uni}}\right)\right] + \mathbb{E}\left[R\left(A_{\text{greedy}}\right)\right], \tag{6.5}$$

which is the expected reward of the exploration phase plus the expected reward of the exploitation phase.

In what follows, we derive lower bounds for $\mathbb{E}\left[R\left(A_{\text{uni}}\right)\right]$ and $\mathbb{E}\left[R\left(A_{\text{greedy}}\right)\right]$ independently. Then, putting these together, we have a lower bound for the expected reward of $A_{\epsilon-\text{first}}$. Following this, we derive an upper bound for the expected reward of the optimal sequence $A^*$. The difference between the lower bound of $\mathbb{E}\left[R\left(A_{\epsilon-\text{first}}\right)\right]$ and the upper bound of $\mathbb{E}\left[R\left(A^*\right)\right]$ then gives us the upper bound of the regret function of our proposed algorithm. However, this bound is constructed using Hoeffding's inequality, so it correct only with a certain probability. Specifically, it is correct with probability $(1-\beta)^K$, where $\beta \in (0,1)$ is a predefined confidence parameter (i.e. the confidence with which we want the upper bound to hold) and $k$ is the number of arms.

Given this, without loss of generality, for ease of exposition we assume that the reward distribution of each arm has support in $[0,1]$, and the pulling cost $c_i > 1$ for each $i$ (our result can be scaled for different size supports and costs as appropriate). We now define some terms. Let

$$I^{\max} = \arg\max_i \frac{\hat{\mu}_i}{c_i} \tag{6.6}$$

denote the arm with the highest mean value density estimate. Similarly, let $i^{\max}$ and $i^{\min}$ denote the arm with the highest and the lowest real mean value densities, respectively.

That is, we have:

$$i^{\max} = \arg\max_i \frac{\mu_i}{c_i} \tag{6.7}$$

$$i^{\min} = \arg\min_i \frac{\mu_i}{c_i} \tag{6.8}$$

Then, let $D_{\max}$ denote the difference between the highest and the lowest mean value densities; that is:

$$D_{\max} = \frac{\mu_{i^{\max}}}{c_{i^{\max}}} - \frac{\mu_{i^{\min}}}{c_{i^{\min}}} \tag{6.9}$$

Finally, let $A_{\text{arb}}$ be an arbitrary exploration sequence of pulls. We say that $A_{\text{arb}}$ exploits the budget dedicated to the exploration if and only if after $A_{\text{arb}}$ stops, none of the arms can be additionally pulled without exceeding the exploration budget. Given this, we now state the main contribution of the paper.

**Theorem 6.1.** *Suppose that $A_{\text{arb}}$ is an arbitrary exploration sequence, that exploits the given exploration budget, and that within this sequence, each arm $i$ is pulled $n_i$ times. Let $A$ denote the joint sequence of $A_{\text{arb}}$ and $A_{\text{greedy}}$, where $A_{\text{greedy}}$ is as defined in section 6.1.2.2. Then, for any $k > 1$, $B > 0$, and $0 < \epsilon, \beta < 1$, with probability $(1-\beta)^K$, the regret function of sequence $A$ is at most:*

$$2 + \epsilon B D_{\max} + B\left(\sqrt{\frac{-\ln\beta}{n_{i^{\max}}}} + \sqrt{\frac{-\ln\beta}{n_{I^{\max}}}}\right)$$

*where $n_{i^{\max}}$ and $n_{I^{\max}}$ are the number of pulls of arms $i^{\max}$ and $I^{\max}$, respectively.*

Before we prove this theorem, let us prove the following auxiliary lemmas.

**Lemma 6.2.** *Let $A_{\text{arb}}$ denote an arbitrary exploration sequence that exploits its exploration budget. Within this sequence, each arm $i$ is pulled $n_i$ times. Thus, we have:*

$$\sum_{i=1}^{K} n_i \mu_i \geq \frac{\epsilon B \mu_{i^{\min}}}{c_{i^{\min}}} - 1$$

**Lemma 6.3.** *If $A_{\text{greedy}}$ is the density-ordered greedy exploitation algorithm, then*

$$\mathbb{E}\left[R\left(A_{\text{greedy}}\right)\right] \geq (1-\epsilon)\frac{B\mu_{I^{\max}}}{c_{I^{\max}}} - 1$$

**Lemma 6.4.** *If $A^*$ is the optimal sequence defined in equation (6.2), then*

$$\mathbb{E}\left[R\left(A^*\right)\right] \leq \frac{B\mu_{i^{max}}}{c_{i^{max}}}$$

**Lemma 6.5.** *If $|a-b| \leq \delta_1$, $|c-d| \leq \delta_2$, and $a \geq c$, then $d \leq b + \delta_1 + \delta_2$*

**Proof of lemma 6.2**. If $A_{\text{arb}}$ exploits the budget for exploration, it is true that for any $c_j$, $\sum_{i=1}^{K} n_i c_i \geq \epsilon B - c_j$, since none of the arms can be pulled after the stop of $A_{\text{arb}}$, without exceeding $\epsilon B$. Thus, we have:

$$\begin{aligned}\sum_{i=1}^{K} n_i \mu_i &= \sum_{i=1}^{K} n_i c_i \frac{\mu_i}{c_i} \geq \left(\sum_{i=1}^{K} n_i c_i\right) \frac{\mu_{i^{\min}}}{c_{i^{\min}}} \\ &\geq (\epsilon B - c_{i^{\min}}) \frac{\mu_{i^{\min}}}{c_{i^{\min}}} \geq \frac{\epsilon B \mu_{i^{\min}}}{c_{i^{\min}}} - 1\end{aligned}$$

since $\mu_i \leq 1$ for $\forall i$.

□

**Proof of lemma 6.3**. By just pulling arm $I^{\max}$ in the exploitation phase, which is the first round of $A_{\text{greedy}}$, the expected reward we can get there is $\lfloor \frac{(1-\epsilon)B}{c_{I^{\max}}} \rfloor \mu_{I^{\max}}$, where $\lfloor \frac{(1-\epsilon)B}{c_{I^{\max}}} \rfloor$ is the number of pulls of arm $I^{\max}$. Thus, we have:

$$\begin{aligned}\mathbb{E}\left[R\left(A_{\text{greedy}}\right)\right] &\geq \left\lfloor \frac{(1-\epsilon)\,B}{c_{I^{\max}}} \right\rfloor \mu_{I^{\max}} \\ &\geq \left(\frac{(1-\epsilon)\,B}{c_{I^{\max}}} - 1\right) \mu_{I^{\max}} \geq (1-\epsilon)\,\frac{B\mu_{I^{\max}}}{c_{I^{\max}}} - 1\end{aligned}$$

since $\mu_i \leq 1$ for $\forall i$.

□

**Proof of lemma 6.4**. Suppose that in the optimal sequence, $N_i$ is the total number of pulls of arm $i$. Thus, the cost constraint can be formulated as $\sum_{i=1}^{K} N_i c_i \leq B$. Given this, we have:

$$\mathbb{E}\left[R\left(A^*\right)\right] = \sum_{i=1}^{K} N_i \mu_i = \sum_{i=1}^{K} N_i c_i \frac{\mu_i}{c_i} \leq \left(\sum_{i=1}^{K} N_i c_i\right) \frac{\mu_{i^{\max}}}{c_{i^{\max}}} \leq \frac{B\mu_{i^{\max}}}{c_{i^{\max}}}$$

□

**Proof of lemma 6.5**. Since $|a-b| \leq \delta_1$, we have $a \leq b + \delta_1$. Similarly, we have $d \leq c + \delta_2$. Since $a \geq c$, we have the following: $d \leq c + \delta_2 \leq a + \delta_2 \leq b + \delta_1 + \delta_2$.

□

**Proof of theorem 6.1**. Let $n_i$ denote the number of pulls of arm $i$ in the exploration phase. Using Hoeffding's inequality for each arm $i$, we have:

$$P\left(|\hat{\mu}_i - \mu_i| \geq \delta_i\right) \leq \exp\left\{-n_i \delta_i^2\right\}$$

that is, dividing by $c_i$, we have:

$$P\left(\left|\frac{\hat{\mu}_i}{c_i}-\frac{\mu_i}{c_i}\right|\geq\frac{\delta_i}{c_i}\right)\leq\exp\left\{-n_i\delta_i^2\right\}$$

which is equivalent to the following:

$$P\left(\left|\frac{\hat{\mu}_i}{c_i}-\frac{\mu_i}{c_i}\right|\geq\delta_i\right)\leq\exp\left\{-n_i\delta_i^2c_i^2\right\} \tag{6.10}$$

Setting $\delta_i=\sqrt{\frac{-\ln\beta}{n_ic_i^2}}$, equation (6.10) can be reformulated as follows:

$$P\left(\left|\frac{\hat{\mu}_i}{c_i}-\frac{\mu_i}{c_i}\right|\geq\delta_i\right)\leq\beta$$

Thus, with probability $(1-\beta)^K$, for each arm $i$, we have:

$$\left|\frac{\hat{\mu}_i}{c_i}-\frac{\mu_i}{c_i}\right|\leq\delta_i \tag{6.11}$$

Hereafter, we strictly focus on the case, whereby equation (6.11) holds for each arm $i$. The reward collected in the exploration phase can be calculated as follows:

$$\mathbb{E}\left[R\left(A_{\text{arb}}\right)\right]=\sum_{i=1}^{K}n_i\mu_i\geq\frac{\epsilon B\mu_{i^{\min}}}{c_{i^{\min}}}-1 \tag{6.12}$$

The right side of equation (6.12) holds, due to lemma 6.2. Using lemma 6.3 and equation (6.12), we get the following:

$$\begin{aligned}\mathbb{E}\left[R\left(A\right)\right] &= \mathbb{E}\left[R\left(A_{\text{arb}}\right)\right]+\mathbb{E}\left[R\left(A_{\text{greedy}}\right)\right]\\ &\geq \frac{\epsilon B\mu_{i^{\min}}}{c_{i^{\min}}}-1+\left(1-\epsilon\right)\frac{B\mu_{I^{\max}}}{c_{I^{\max}}}-1\\ &\geq \frac{\epsilon B\mu_{i^{\min}}}{c_{i^{\min}}}+\left(1-\epsilon\right)\frac{B\mu_{I^{\max}}}{c_{I^{\max}}}-2\end{aligned} \tag{6.13}$$

By denifition, we have $\frac{\mu_{I^{\max}}}{c_{I^{\max}}}\geq\frac{\mu_{i^{\max}}}{c_{i^{\max}}}$, and $\frac{\hat{\mu}_{i^{\max}}}{c_{i^{\max}}}\geq\frac{\hat{\mu}_{I^{\max}}}{c_{I^{\max}}}$. Furthermore, equation (6.11) holds for each arm $i$. According to lemma 6.5, we have:

$$\frac{\mu_{I^{\max}}}{c_{I^{\max}}}\geq\frac{\mu_{i^{\max}}}{c_{i^{\max}}}-\delta_{i^{\max}}-\delta_{I^{\max}} \tag{6.14}$$

Substituting this into equation (6.13), we have:

$$\begin{aligned}\mathbb{E}\left[R\left(A\right)\right] &\geq \frac{\epsilon B\mu_{i^{\min}}}{c_{i^{\min}}}+\left(1-\epsilon\right)B\left(\frac{\mu_{i^{\max}}}{c_{i^{\max}}}-\delta_{i^{\max}}-\delta_{I^{\max}}\right)-2\\ &\geq \frac{\epsilon B\mu_{i^{\min}}}{c_{i^{\min}}}+B\frac{\mu_{i^{\max}}}{c_{i^{\max}}}-\frac{\epsilon B\mu_{i^{\max}}}{c_{i^{\max}}}-\\ &\quad-\left(1-\epsilon\right)B\left(\delta_{i^{\max}}+\delta_{I^{\max}}\right)-2\end{aligned} \tag{6.15}$$

According to lemma 6.4, $\mathbb{E}[R(A^*)] \leq \frac{B\mu_{i^{\max}}}{c_{i^{\max}}}$. Thus, by substituting it into equation (6.15), we have:

$$\begin{aligned}\mathbb{E}[R(A)] \quad \geq \quad & \mathbb{E}[R(A^*)] + \frac{\epsilon B\mu_{i^{\min}}}{c_{i^{\min}}} - \frac{\epsilon B\mu_{i^{\max}}}{c_{i^{\max}}} - \\ & - (1-\epsilon) B(\delta_{i^{\max}} + \delta_{I^{\max}}) - 2 \end{aligned} \tag{6.16}$$

Using the definition of regret function, defined in equation (6.3), and that $(1-\epsilon) < 1$, we have the following upper bound for $L(A)$:

$$L(A) \leq 2 + \epsilon B D_{\max} + B(\delta_{i^{\max}} + \delta_{I^{\max}}) \tag{6.17}$$

where $D_{\max} = \frac{\mu_{i^{\max}}}{c_{i^{\max}}} - \frac{\mu_{i^{\min}}}{c_{i^{\min}}}$. Thus, by replacing $\delta_i = \sqrt{\frac{-\ln\beta}{n_i c_i^2}}$ for $i = I^{\max}$ and $i = i^{\max}$, and using the fact that $c_i \geq 1$ for each $i$, we get:

$$L(A) \quad \leq \quad 2 + \epsilon B D_{\max} + B\left(\sqrt{\frac{-\ln\beta}{n_{i^{\max}}}} + \sqrt{\frac{-\ln\beta}{n_{I^{\max}}}}\right)$$

□

Note that this theorem is of no practical use, since it assumes knowledge of $i^{\max}$ and $i^{\min}$, which is unlikely in real applications. However, it gives us a generic upper bound for the regret function that can be refined for specific exploration methods.

In particular, we now refine this upper bound for the case of uniform pull exploration $A_{\text{uni}}$. Since we uniformly pull the arms, $A_{\text{uni}}$ exploits the exploration budget. Furthermore, according to Equation 6.4, each arm is pulled at least $\lfloor \epsilon B / \sum_{j=1}^{K} c_j \rfloor$ times. Thus, we can state the following:

**Corollary 6.6.** *Let $A_{\epsilon-\text{first}}$ denote the pulling sequence of the $\epsilon$-first algorithm with uniform pull exploration and density value–ordered greedy exploitation. For any $k > 1$, $B > 0$, and $0 < \epsilon, \beta < 1$, with probability $(1-\beta)^K$, regret function of $A_{\epsilon-\text{first}}$ is at most*

$$L(A_{\epsilon-\text{first}}) \leq 2 + \epsilon B D_{\max} + 2B\left(\sqrt{\frac{(-\ln\beta)}{\lfloor \epsilon B / \sum_{j=1}^{K} c_j \rfloor}}\right)$$

Apart from the uniform pull algorithm, other exploration policies, such as UCB, or Boltzmann exploration [Cicirello and Smith, 2005], can be applied within our $\epsilon$-first approach. The upper bound given in Theorem 6.1 also holds for those policies. However, refining this upper bound, as we did in the case of the uniform pull algorithm, is a challenge. In particular, estimating the number of times each arm is pulled is difficult in both UCB and Boltzmann explorations. Thus, refined upper bounds cannot be derived for these policies using the same technique as for the uniform pull policy. As

such, we must rely on empirically comparisons of $\epsilon$–first with uniform and non–uniform exploration policies.

### 6.1.4 Performance Evaluation

In this section, we evaluate our $\epsilon$–first algorithm with uniform exploration and with UCB–based exploration, and compare them to a modified version of the $\epsilon_n$*–greedy* algorithm. The latter was originally proposed to solve the standard MAB problem, while UCB–exploration was developed to solve the MAB with pure exploration. We compare these algorithms in three scenarios, which highlight the benefits of using an algorithm that is tailored to the budget–limited MAB. However, before discussing the experiments, we first describe the benchmark algorithms and explain why they were chosen.

The $\epsilon_n$–greedy algorithm is designed for standard MABs. It learns the values of all arms while still selecting the arm with the highest reward infinitely more frequently than any other, and can be described as follows: At each time–step $n$, the arm with the *highest expected reward estimate* is pulled with probability $1-\epsilon_n$, otherwise a randomly selected arm is pulled. The value of $\epsilon_n$ is decreased over time, proportional to $O\left(1/n\right)$, starting from an initial value $\epsilon_0$. However, since $\epsilon_n$–greedy does not take pulling cost into account, we modify it to fit our model's cost constraints. Specifically, the arm with the highest reward value–cost ratio is pulled with probability $1-\epsilon_n$, with any other randomly chosen arm pulled with probability $\epsilon$. Now, consider the case of a budget–limited MAB when the budget $B$ is considerably larger than the pulling cost of each arm (e.g. $B \to \infty$). In this situation, the difference between the pulling cost of the arms are not significant, and thus, this version of our problem could be approximated by a standard MAB (without pulling cost). However, when $B$ is comparable to the arms' pulling costs, it is not obvious how traditional MAB algorithms, such as $\epsilon_n$-greedy, behave.

Conversely, since our proposed $\epsilon$-first approach relies significantly on the efficiency of the exploration phase, it is not obvious that uniformly pulling the arms is the best choice for exploration. Thus, there is a need to compare our proposed algorithm with $\epsilon$-first approaches that use different exploration techniques. Given this, we choose an $\epsilon$*-first approach with UCB–based exploration* as the second benchmark algorithm. The UCB exploration phase pulls the arm with the highest upper confidence bound on the arm's expected value, in order to determine the arm with the highest expected reward (see section 2.3 for more details). Since UCB was proposed for MAB with non–costly arms, we also have to modify its goal, similarly to the case of $\epsilon_n$-greedy. Note that, according to Bubeck *et al.* [2009a], UCB typically results in better performance in exploration than the uniform pull approach.

Now, recall that due to the limited budget, the optimal solution of our MAB problem is not to repeatedly pull the optimal arm, but to pull the optimal combination of arms.

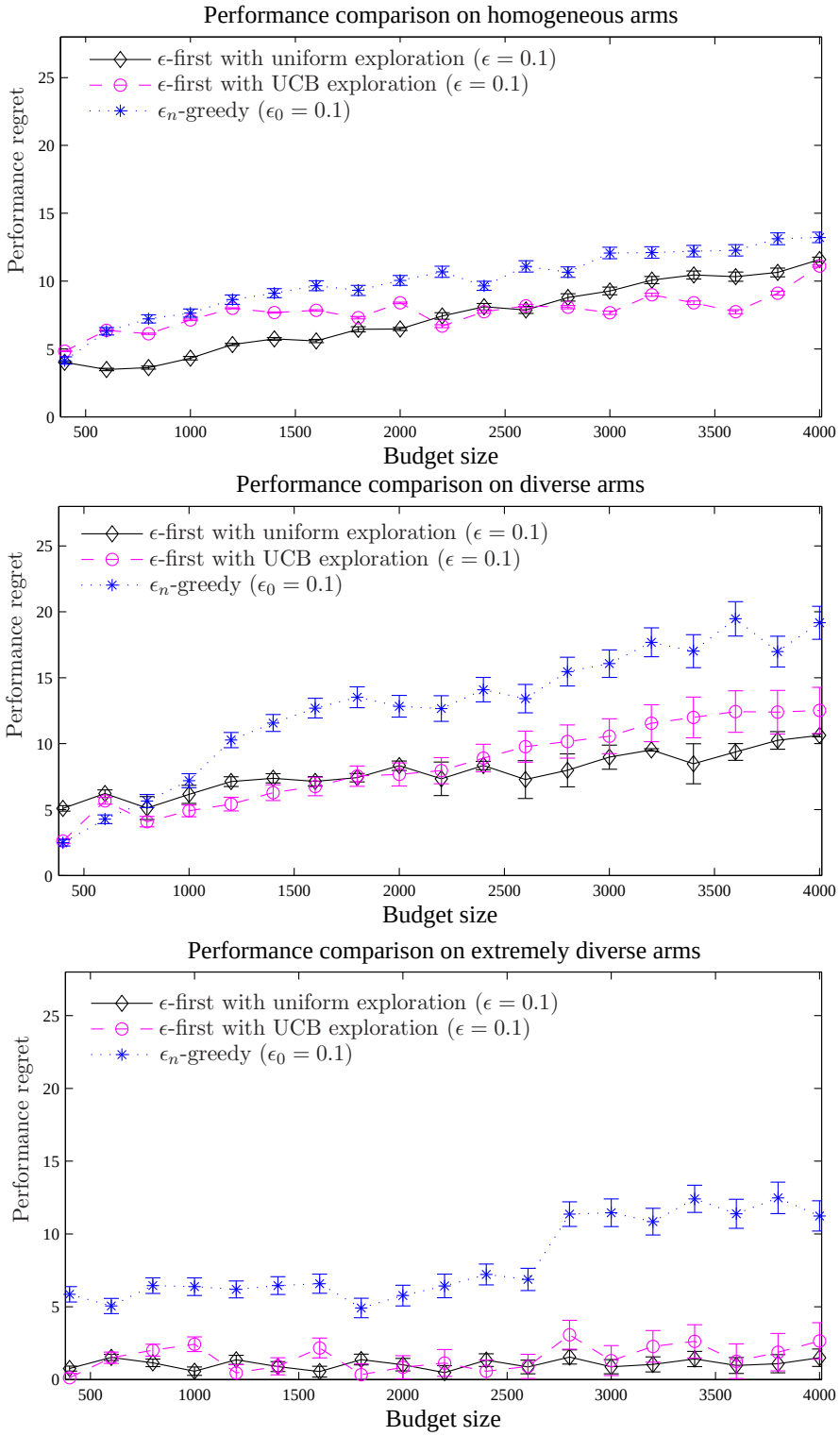



FIGURE 6.1: Performance of the algorithms in the case of 6-armed bandit machine, which has: (A) homogeneous arms with expected reward value–cost parameter pairs $\{4,4\}$, $\{4,4\}$, $\{4,4\}$, $\{2.7,3\}$, $\{2.7,3\}$, $\{2.7,3\}$; (B) diverse arms with parameter pairs $\{3,4\}$, $\{2,4\}$, $\{0.2,2\}$, $\{0.16,2\}$, $\{18,20\}$, $\{18,16\}$; and (C) extremely diverse arms with with parameter pairs $\{0.44,5\}$, $\{0.4,4\}$, $\{0.2,3\}$, $\{0.08,1\}$, $\{14,120\}$, $\{18,150\}$.

However, our proposed exploitation algorithm, the reward–cost ratio ordered greedy method, does not always return an optimal combination, because it is an approximate algorithm. Specifically, if the arms are homogeneous, that is, the pulling costs do not significantly differ from each other, then the reward–cost ratio ordered greedy typically results in solely pulling the arm with highest reward estimate–cost ratio. Consequently, it shows similarities to the $\epsilon_n$–greedy policy in this case. On the other hand, when the arms are more diverse, that is, the pulling costs are significantly different, the algorithm pulls more arms (for more details see [Kohli *et al.*, 2004]).

Given this, in order to measure the efficiency of the reward–cost ratio ordered greedy in different situations, we set three test cases, each of which considers a 6-armed bandit machine which has: (i) homogenous arms; (ii) moderately diverse arms; and (iii) extremely diverse arms. In the moderately and extremely cases, two out of six arms have moderately/extremely more expensive pulling cost than the others. Given this, the expected reward and cost values were randomly chosen, with regard to these diversity properties. Specifically, the distribution of the rewards are Gaussian, with means given as the first element of the expected reward value–cost pairs in the caption of Figure 6.1, with 0.1 variance value , and supports $[0, 20]$. The cost of each arm is given as the second element of the expected reward value–cost pairs. Finally, for all test cases, we run our simulation with different budget values, from 400 to 4000. The numerical results are depicted in Figure 6.1. Within this figure, the error bars represent the 95% confidence intervals.

From Figure 6.1, we can see that in the homogeneous case, both $\epsilon_n$–greedy and the $\epsilon$–first with uniform exploration approach achieve similar efficiency. However, our algorithm results in better performance when the arms are more diverse. In particular, the performance regret of our algorithm is 50% less in the diverse case, and 80% less in the extremely diverse case, than that of the $\epsilon_n$–greedy. In addition, despite the better performance of UCB in exploration, the performance of our algorithm surprisingly does not significantly differ from that of the $\epsilon$–first with UCB–based exploration. This is due to both uniform and UCB–based explorations can efficiently determine those arms which are pulled in the reward–cost ratio ordered greedy algorithm.

In order to show this, and to highlight the differences of each algorithm's behaviour as well, consider a single run of the case of extremely diverse arms, with budget $B = 3500$. Figure 6.2 depicts the number of pulls of each particular arm. Here, the optimal solution is to pull arm 2 twelve times, arm 4 two times, and arm 6 twenty–three times. We can see that both $\epsilon$–first with uniform and with UCB–based exploration show similar behaviour to the optimal solution, however, $\epsilon_n$–greedy focuses on arms 2 and 5 instead, while arm 6 is not sampled at all. The reason is, due to its nature, if $\epsilon_n$–greedy starts with a wrong arm (i.e. not the best arm), it will stick with it for a while. Therefore, in order to learn

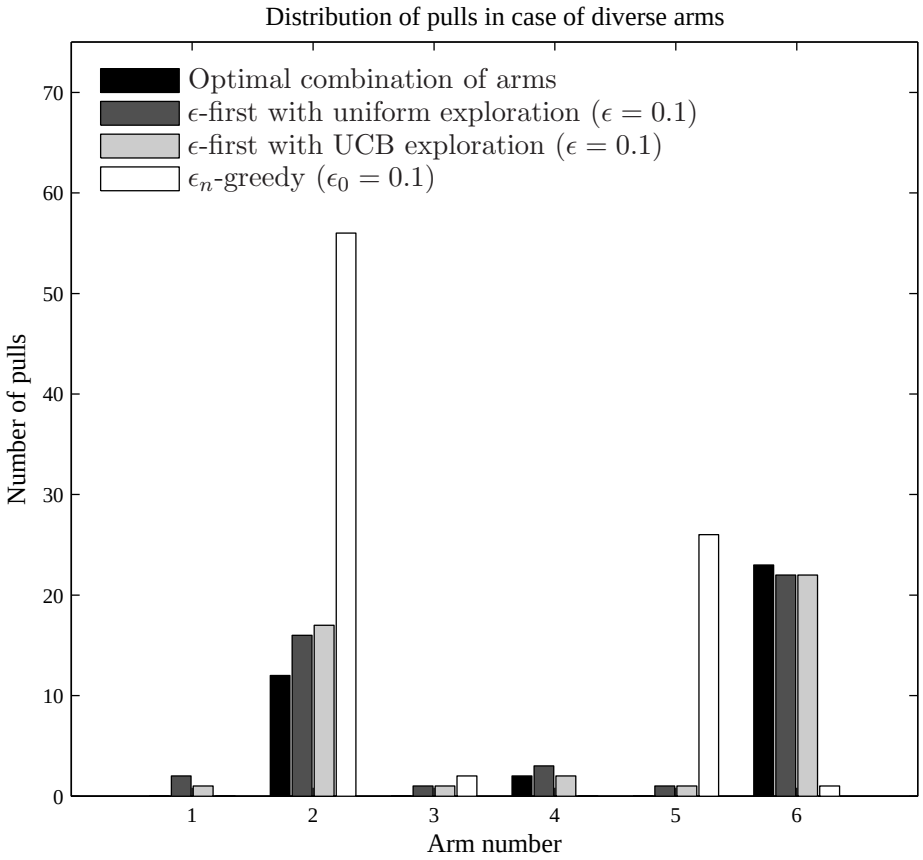



FIGURE 6.2: Number of pulls of each arm in the case of 6–armed bandit with extremely diverse arms, and budget $B = 3500$.

the best combination, it needs more time. However, due to the limited budget, it does not have enough time to do this.

## 6.2 Budgeted Multi–Armed Bandits with Pulling Cost for Energy Management

As mentioned earlier, in this section, we study the energy management problem with non–harvesting sensors. In particular, within this problem, it is not possible for the agents to harvest energy from the environment (see section 3.3.2 for more details). Given this, in order to tackle this problem, we use the the formal model and the notations introduced in section 3.3.2.

Recall that within this model, each agent $i$ has an energy budget $B_i^{\mathrm{T}}(t)$ for each time slot $t$, where $B_i^{\mathrm{T}}(t)$ is the total residual energy level of the battery. That is, each agent has to rely on its battery level in the whole operation time. The value of $B_i^{\mathrm{T}}(t)$ is updated by using equation 3.10. Since at each time slot, the agents have to allocate

energy to the tasks of sampling, receiving, and transmitting data packets, the goal here is to efficiently allocate the energy budgets to these tasks, such that the total energy consumption does not exceed the total energy budget.

Against this background, we use a budgeted MAB model with pulling cost to tackle this energy management problem. More precisely, we first demonstrate that it is not possible to use the standard MAB model to describe this problem, as we did in the case of energy–harvesting sensors. Following this, we show how to model this problem as a budgeted MAB with pulling cost model.

In so doing, we first define the reward function and the action set of each agent, in order to formulate a MAB model. Since theorem 5.4 does not assume anything related to the capability of energy–harvesting, it holds for the case of non–harvesting sensors as well. Given this, for the reward function, we can use the reward function defined in equation 5.9. Similarly, to define a single action of an agent, we can use definition 5.3 of section 5.2. That is, each agent's single action can be described with a 3-tuple $a_i(t) = \left\langle n_i^{\mathrm{S}}(t), n_i^{\mathrm{Rx}}(t), n_i^{\mathrm{Tx}}(t) \right\rangle$, where $n_i^{\mathrm{S}}(t)$, $n_i^{\mathrm{Rx}}(t)$, and $n_i^{\mathrm{Tx}}(t)$ denote the capacity of sampling, receiving, and transmitting, respectively. However, since in the case of non–harvesting sensors, the budget limit for each time slot is not fixed, we have the following constraint:

$$n_i^{\mathrm{S}}(t)\, e_i^{\mathrm{S}} + n_i^{\mathrm{Rx}}(t)\, e_i^{\mathrm{Rx}} + n_i^{\mathrm{Tx}}(t)\, e_i^{\mathrm{Tx}} \leq B_i^{\mathrm{T}}(t) \tag{6.18}$$

Since $B_i^{\mathrm{T}}(t)$ varies over time, the set of actions of each agent varies ovet time as well. This indicates that standard MAB models do not fit this problem of energy management. On the other hand, we can formulate this problem as a budgeted MAB with pulling cost as follows.

Note that the agents consumes energy when it allocates energy budgets to its tasks of sampling, receiving, and transmitting. Since an action of the agent is to allocate energy to these tasks, the actions here can be regarded as costly. Furthermore, the overall budget of an agent is the initial energy level of the agent's battery. Given this, both the exploration and exploitation phases of the agent is limited by that overall budget. Thus, this problem can be regarded as a budgeted MAB with pulling cost.

Since the budgeted $\epsilon$–first approach, introduced in section 6.1.2, is the first algorithm developed for the problem of budgeted MAB with pulling cost, hereafter we focus on the use of this approach for our energy management problem. Given this, we analyse the performance of this approach in the next section. Furthermore, we also highlight on its advantages and drawbacks as well.

## 6.3 Performance Analysis

In this section, we focus on the performance analysis of the budgeted $\epsilon$–first approach for our energy management problem. Specifically, we analyse its regret bound in the network level (i.e. taking all the agents into account). Furthermore, we highlight its major advantages and disadvantages. Given this, section 6.3.1 discusses the regret bound of the network's performance. In addition, section 6.3.2 studies the advantages and disadvantages of MAB/EM against the research requirements given in section 1.1.

### 6.3.1 Regret Bounds on the Performance

As mentioned in section 6.1.3, budgeted $\epsilon$–first approach with uniform pull exploration has a practical regret upper bound for its performance (see corollary 6.6 for more details). Given this, we analyse the performance of the network of agents in which each agent uses budgeted $\epsilon$–first approach with uniform pull exploration for energy management.

Now, in the vein of equation 6.9, let $D^i_{\max}$ denote the difference between the highest and the lowest mean value densities of agent $i$. Let $K^i$ denote the number of actions agent $i$ can choose (i.e. the total number of energy budget allocation combinations of agent $i$). Furthermore, let $c^i_j$ denote the cost of agent $i$'s action $j$ (i.e. the total energy allocated to the sensory tasks when agent $i$ chooses allocation combination $j$). Finally, let $B^i$ denote the initial total energy budget of agent $i$'s battery. According to corollary 6.6, for each agent $i$, we have the following upper bound for the performance regret:

$$L_i\left(A_{\epsilon-\text{first}}\right) \leq 2 + \epsilon B^i D^i_{\max} + 2B^i \left(\sqrt{\frac{(-\ln \beta)}{\lfloor \epsilon B^i / \sum_{j=1}^{K^i} c^i_j \rfloor}}\right) \tag{6.19}$$

where $L_i\left(A_{\epsilon-\text{first}}\right)$ is the regret function of agent $i$, by using the budgeted $\epsilon$–first approach with uniform pull exploration.

Given this, if $N$ denotes the number of agents in the network, then for the upper bound of the total performance regret of the network, we have:

$$\sum_{i=1}^{N} L_i\left(A_{\epsilon-\text{first}}\right) \quad \leq \quad 2N + \sum_{i=1}^{N} \left\{ \epsilon B^i D^i_{\max} + 2B^i \left(\sqrt{\frac{(-\ln \beta)}{\lfloor \epsilon B^i / \sum_{j=1}^{K^i} c^i_j \rfloor}}\right) \right\} \tag{6.20}$$

That is, the total performance regret of the network is bounded by equation 6.20.

### 6.3.2 Advantages and Shortcomings

From the aspect of computational complexity, energy management with the budgeted $\epsilon$–first approach is simple. In particular, since we focus on uniform pull exploration,

the total computational complexity of each agent $i$ is $O\left(K^i \ln K^i\right)$ in total, where $K^i$ is the size of agent $i$'s action set. This value of complexity is calculated as follows. In the exploration phase, the uniform pull policy simply pulls the next arm, that is, the complexity is $O(1)$. In the exploitation phase, the agent uses the reward–cost ratio ordered greedy algorithm, which has the complexity of $O\left(K^i \ln K^i\right)$ in total (see section 6.1.2.2 for more details).

In addition, similarly to MAB/EM, here the agents do not have to communicate to each other, in order to coordinate their actions in energy management (see section 5.3.2 for more details). However, since the budgeted $\epsilon$–first approach explicitly separates the exploration from the exploitation phase by first exploring and then exploiting, this approach is not efficient in dynamic environments. Specifically, by having the exploration phase before exploitation, the agent can only learn the initial characteristics of the environment. Thus, if environmental changes occur during the exploitation phase (i.e. after the exploration), the agent cannot determine those, and thus, it cannot adopt to the changes. Given this, budgeted $\epsilon$–first approach cannot perform well in the WSN domain, since it cannot adopt to the environmental changes. Furthermore, to date, none of other approaches has been proposed for the dynamic case of the budgeted MAB problem with pulling cost. Due to these reasons, at the moment, we do not implement any algorithms to the budgeted MAB based energy management approach. However, in the future work, we aim to develop such algorithms for the problem of budgeted MAB with pulling cost, that are efficient in dynamic environments as well. These algorithms then can be used for tackling the energy management problem with non–harvesting sensor nodes.

## 6.4 Summary

In this chapter, we have proposed a MAB learning based energy management approach for the case of non–harvesting sensor agents. Since none of the existing MAB models can fit this problem, we have introduced a new version of MAB, the budgeted MAB with pulling cost, in order to tackle the energy management problem. In this model, the number of pulls is limited, and the pulls are costly. Thus, different pulling behaviour is needed within this model, in order to achieve maximal expected total reward. Given this, we have proposed an $\epsilon$-first based approach that splits the overall budget into exploration and exploitation budgets. Our algorithm uses a uniform pull policy for exploration, and the reward–cost ratio ordered greedy algorithm for exploitation. Beside this, we have devised a theoretical upper bound for the regret function of generic $\epsilon$–first approaches with arbitrary exploration policies. We also have refined this upper bound to the specific case of uniform pull exploration. Finally, through simulation, we have

demonstrated that the $\epsilon$–first approaches outperform the $\epsilon_n$–greedy method, an efficient algorithm for standard MAB problems.

Following this, we have introduced devised the budgeted $\epsilon$–first approach for this problem. In more detail, we have proposed a uniform pull exploration and reward reward–cost ratio ordered exploitation policies. In addition, we have analysed the performance of the budgeted $\epsilon$–first approach, by proposing an upper bound for the performance of the approach. This upper bound is generic in the following sense: it holds for any arbitrary exploration policy. We then have refined the upper bound for the case of uniform pull exploration. Following this, we have also analysed the performance of the proposed approach by running extensive simulations. Here, we have demonstrated that traditional MAB approaches are not suitable for the new MAB problem, since their performance is outperformed by that of the $\epsilon$–first approach. Besides, although $\epsilon$–first approach with uniform exploration does not perform well in every cases, we have shown that the performance of the uniform exploration is as good as that of the UCB–based exploration.

Based on the budgeted MAB with pulling cost model, we have developed a MAB based approach for the energy management problem with non–harvesting sensors. Since to date, only the budgeted $\epsilon$–first approach is proposed for the budgeted MAB with pulling cost problem, we have used this approach to tackle the energy management problem. However, this approach is not suitable for dynamic environments, since it only explores at the beginning. Given this, new approaches are needed to develop in the future, in order to efficiently deal with the dynamism of the environment.

# Chapter 7

# Conclusions and Future Work

In this chapter, we present a global view on the contributions of this report towards developing efficient long–term information collection policies for wireless sensor networks. To begin, in section 7.1, we first summarise the research carried out in order to achieve this goal. In so doing we also explain how we satisfied each of the research requirements that we initially set out at the beginning of this report. Then, in section 7.2, we outline some general areas of future work that follow from this report.

## 7.1 Summary of Results

In this report, we have studied long-term information collection in the WSN domain. In particular, first we have looked at previous work on adaptive sampling, adaptive routing, information processing and energy management in WSNs. These are the perspectives from which efficient information collection can be achieved. Following this, we have introduced our model for WSNs that includes the aforementioned perspectives, and we have formalised our main objective of maximising the total amount of collected information at the base stations over a given finite time interval.

In a review of the relevant literature, we have shown that none of the state-of-the-art algorithms in the aforementioned perspectives meets our research requirements for long-term information collection. Against this background, we have developed a novel model that enables learning and adopting to the environmental changes. We assumed that the data sampling and information content valuation parts of the model use state-of-the-art techniques. On the other hand, we have focused specifically on the routing and energy management parts. Given this, we have decomposed our main objective into two parts, one for the routing and one for the energy management. Given this, we have decomposed the original problem into two sub–problems, namely: (i) the maximal information throughput routing problem; and (ii) the energy management problem. For

the former, we have proposed the maximal information throughput routing algorithm (MITRA), a decentralised method that provides the optimal solution for the maximal information throughput routing problem (contribution 1). We also have provided a proof of the optimality of the MITRA. In order to do this, we first solved the routing problem in a centralised way, by using graph theory approach. Following this, based on the results of the solution for the centralised case, we proposed a decentralised algorithm, whose optimality has been proven as well. Moreover, by comparing its performance to a greedy and naïve algorithm, we have shown that this algorithm has low communication and computational costs, compared to the size of the network.

Following this, we have focused on the energy management problem. Here, we have identified two cases, namely: (i) energy management with energy–harvesting sensors; and (ii) energy management with non–harvesting sensors. For the first case, we have developed a multi–armed bandit based approach called MAB/EM (contribution 2). In particular, we first have introduced the non–stochastic MAB model, which forms the foundation of our approach. Then we have reduced the energy management problem into a non–stochastic MAB problem, by defining the actions and the reward functions for the agents. Following this, we have described in more detail a state-of-the-art MAB algorithm, the Exp3, that can be used to efficiently deal with the energy management problem. To measure the efficiency of this MAB based approach, we have compared the performance of the MAB/EM with a state–of–the–art non–learning information collection algorithm, USAC. Moreover, to measure the (weak) performance regret of MAB/EM, we have also compared its performance with the centralised best fixed policy algorithm, which is centralised and uses global information in order to calculate the optimal solution. Both comparisons showed that MAB/EM and MITRA together are efficient in terms of long–term information collection, since it can adopt to the environmental changes, and outperforms the other state–of–the–art non–learning algorithms. Furthermore, it has low computational and communication cost as well.

For the case of energy management with non–harvesting sensors, we have shown that existing MAB models are not suitable to formulate the problem. Given this, we have introduced a new MAB model, the budgeted MAB with pulling cost (contribution 3). In order to efficiently tackle this problem, we have proposed the budgeted $\epsilon$–first approach. In particular, we have proposed a uniform pull method for exploration, and the reward–cost ratio ordered algorithm for exploitation. Following this, we have devised an upper bound of the performance regret for this approach (contribution 4). This bound holds for any arbitrary exploration method. We then have refined this bound for the case of uniform pull exploration. By using extensive simulations, we have shown that existing standard MAB approaches do not perform well in this MAB problem, since our approach outperforms them on average. Based on this MAB model, we have reduced the energy management problem with non–harvesting sensors to a budgeted MAB model

with pulling cost. We also have analysed the performance of the budgeted $\epsilon$–first approach in the WSN domain. We have showed that from the aspect of computational complexity, this approach is simple. However, since the agent first explores and then exploits in the budgeted $\epsilon$–first approach, it cannot perform well in dynamic environments. Thus, it does not fulfill the research requirement of adaptivity. Given this, we aim to develop an algorithm for the budgeted MAB problem with pulling cost, that is adaptive to environmental changes as well.

## 7.2 Future Work

The work described in this report attempts to advance research on information collection in wireless sensor networks, and forms the basis for future work. In particular, we have identified several directions for this future work, ranging from the short–term goals that we will pursue in the upcoming months, to the more general ideas that are considered for the remainder of the time allocated for the PhD. Specifically, the general areas of research that we intend to investigate are:

**Maximal Information Throughput Routing:** In our model, we have assumed a number restrictions, such as the communication is perfect (i.e. no data loss occurs), or the agents can only forward data to other agents that are closer to the $BS$. These restrictions allowed us to simplify the WSN model, and thus, the maximal information throughput problem in the following ways. Due to perfect communication, our proposed algorithm does not have to deal with data loss. That is, it assumes that whenever a packet is transmitted, it will successfully arrive at the receiver. In fact, in real world applications, data loss typically occurs during communication. Furthermore, since data can only be forwarded to nodes that are closer to the $BS$, the routing paths are in fact the shortest paths between the nodes and the $BS$. This simplifies the routing topology of the WSN. However, in some settings it may be sensible for a sensor to initially route data away from the $BS$, if this results in a better route overall. Given this, in the near future, we aim to relax these restrictions, in order to make our WSN model more realistic. That is, we have to modify MITRA so that it can take data loss and non–shortest routing paths into account. In particular, we have to deal with data loss by using techniques that guarantee successful data forwarding, such as retransmission (i.e. data will be repeatedly sent until the receiver receives it), or multi–casting (i.e. multiple copy of the packet is sent to a number of receivers). In addition, the routing algorithm should take into account the case when non–shortest paths are better choices for data forwarding. These tasks will be done in the next three months (see Gantt chart in figure 7.1 for more details).

**Energy Management with Energy–Harvesting Sensors:** In this report, we have assumed that when energy harvesting is possible, the budget of harvested energy is fixed over time. However, due to the dynamic characteristics of the environment of the WSN, it may occur that the harvested energy budget vary over time (e.g. due to different weather conditions in the case of sensors with solar batteries). In this case, the standard MAB model cannot efficiently formulate the energy management problem. In particular, due to the varying harvested energy budget, the action set (i.e. the set of energy budget allocation combinations) of each agent varies over time as well. Thus, the agent not only has to learn the best arm, but to learn the best arms of each different action set. Given this, our goal in the future is to develop a MAB model that fits this type of problem (i.e. MAB with varying action sets). Furthermore, we aim to devise MAB algorithms that can efficiently solve this new MAB problem. In more detail, these algorithms should satisfy the research requirements, namely: (i) adaptivity; (ii) robustness and flexibility; (iii) computational feasibility; and (iv) limited use of communication. Finally, we also aim to devise regret bounds for the performance of the proposed algorithms. These tasks will be carried out after finishing the development of extending routing model with data loss. The total time for studying this problem is approximately three to four months (see figure 7.1).

**Energy Management with Non–Harvesting Sensors:** In the case of energy management with non–harvesting sensors, so far we have introduced the budgeted MAB with pulling cost model for formulating the energy management problem. Furthermore, we have proposed the budgeted $\epsilon$–first approach for dealing with this problem. However, this approach is only suitable for networks with static environments, and it is not efficient in the dynamic case. Given this, our goal is to develop algorithms that can efficiently handle the dynamic environments as well (i.e. it satisfies the research requirement of adaptivity). In more detail, we aim to propose algorithms that have low computational complexity, and work well in environments with limited communication. Furthermore, we also aim to devise regret bounds for the performance of the proposed algorithms. These algorithms then can be the basis of the energy management policy for the case of energy management with non–harvesting sensors. We aim to focus on these tasks after the extension of the energy management model with energy–harvesting sensors. The total time for studying this problem is approximately five months (again, see figure 7.1).

Figure 7.1 summarises our planning of the research activities in the future.

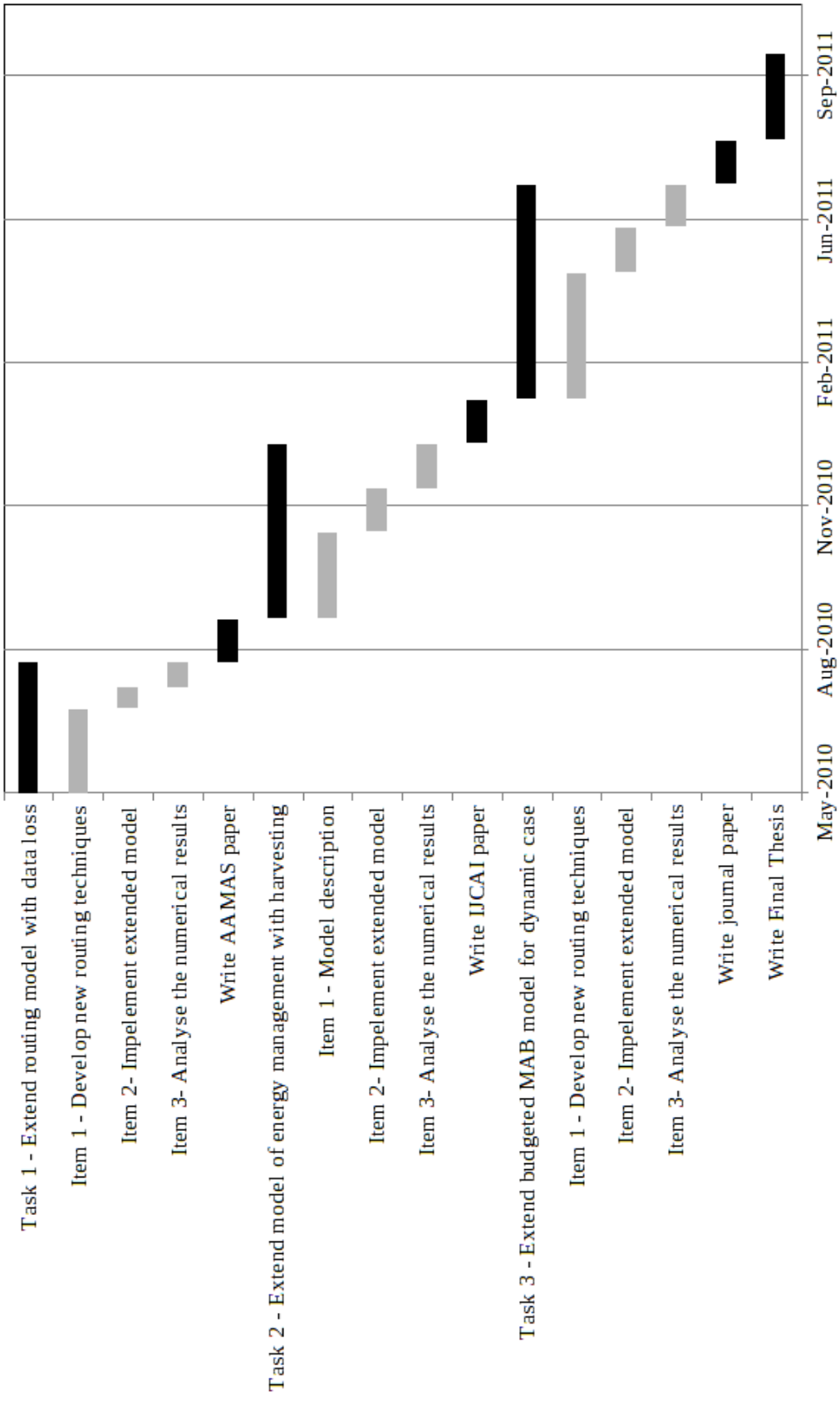



FIGURE 7.1: Gantt chart outlining the planning for future work.

# Bibliography

P. Abbeel, A. Coates, M. Quigley, and A. Y. Ng. An application of reinforcement learning to aerobatic helicopter flight. *In Proceedings of the Twentieth Annual Conference on Neural Information Processing Systems*, pages 1–8, 2006.

R. G. Aghaeil, Md. A. Rahman, W. Gueaieb, and A. El Saddik. Ant colony-based reinforcement learning algorithm for routing in wireless sensor networks. *In Proceedings of the Instrumentation and Measurement Technology Conference*, pages 1–6, 2007.

R. Agrawal. Sample mean based index policies with o($\log n$) regret for the multi-armed bandit problem. *Advances in Applied Probability*, 27:1054–1078, 1995.

F. Ahdi, V. Srinivasan, and K.-C. Chua. Topology control for delay sensitive applications in wireless sensor networks. *Mobile Networks and Applications*, 12(5):406–421, 2007.

K. Akkaya and M. Younis. A survey on routing protocols for wireless sensor networks. *Ad Hoc Networks*, 3(3):325–349, 2005.

I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. Wireless sensor networks: a survey. *Computer Networks*, 38:393–422, 2002.

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

G. Anastasi, M. Conti, A. Falchi, E. Gregori, and A. Passarella. Performance measurements of mote sensor networks. *Proceedings of the ACM/IEEE International Symposium on Modeling, Analysis and Simulation of Wireless and Mobile System*, pages 174–181, 2004.

R. Andonov, V. Poirriez, and S Rajopadhye. Unbounded knapsack problem: dynamic programming revisited. *European Journal of Operational Research*, 123(2):394–407, 2000.

A. Antos, V. Grover, and Cs. Szepesvári. Active learning in multi-armed bandits. *In the Proceedings of the Nineteenth International Conference on Algorithmic Learning Theory*, pages 287–302, 2008.

P. Auer, N. Cesa-Bianchi, and P. Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, 47:235–256, 2002.

P. Auer, N. Cesa-Bianchi, Y. Freund, and R. E. Schapire. The nonstochastic multiarmed bandit problem. *The Society for Industrial and Applied Mathematics Journal on Computing*, 32(1):48–77, 2003.

H. Baldus, K. Klabunde, and G. Muesch. Reliable set-up of medical body-sensor networks. *In Proceedings of the First European Workshop on Wireless Sensor Networks*, pages 353–363, 2004.

Y. Bar-Shalom, X. R. Li, and T. Kirubarajan. *Estimation with Applications to Tracking and Navigation*. Wiley Interscience, 2001. ISBN 047141655X.

S. P. Beeby, M. J. Tudor, and N. M. White. Energy harvesting vibration sources for microsystems applications. *Measurement Science and Technology*, 9:175–195, 2006.

A. Boukerche. *Algorithms and Protocols for Wireless Sensor Networks*. WileyBlackwell, 2008. ISBN : 0471798134.

D. Braginsky and D. Estri. Rumor routing algorithm for sensor networks. *In Proceedings of the ACM International Workshop on Wireless Sensor Networks and Applications*, pages 22–31, 2002.

S. Bubeck, R. Munos, and G. Stoltz. Pure exploration for multi-armed bandit problems. *Algorithmic Learning Theory*, pages 23–37, 2009a.

S. Bubeck, R. Munos, G. Stoltz, and Cs. Szepesvári. Online optimization in x-armed bandits. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, *Advances in Neural Information Processing Systems 21*, pages 201–208. MIT Press, 2009b.

A. R. Cassandra. *Exact and Approximate Algorithms for Partially Observable Markov Decision Processes*. PhD thesis, Brown University, Department of Computer Science, Providence, RI, USA, 1998.

A. Cerpa, J. Elson, D. Estrin, L. Girod, M. Hamilton, and J. Zhao. Habitat monitoring: Application driver for wireless communications technology. *ACM SIGCOMM Workshop on Data Communications in Latin America and the Caribbean*, pages 20–41, 2001.

D. Chakrabarti, R. Kumar, F. Radlinski, and E. Upfal. Mortal multi-armed bandits. *In the Proceedings of Neural Information Processing Systems Conference (NIPS)*, pages 273–280, 2008.

A. Chandrakasan, R. Min, M. Bhardwaj, S. Cho, and A. Wang. Power aware wireless microsensor systems. *Proceedings of Proceeding of the Thirty-Second European Solid-State Device Research Conference*, pages 37–44, 2002.

C.-Y. Chong and S. P. Kumar. Sensor networks: Evolution, opportunities and challenges. *Proceedings of IEEE*, 91(8):1247–1256, 2003.

M. Chu, H. Haussecker, and F. Zhao. Scalable information-driven sensor querying and routing for ad hoc heterogeneous sensor networks. *International Journal of High Performance Computing Applications*, 16(3):293–313, 2002.

V. A. Cicirello and S. F. Smith. The max k-armed bandit: a new model of exploration applied to search heuristic selection. *In the Proceedings of the Twentieth National Conference on Artificial Intelligence (AAAI)*, pages 1355–1361, 2005.

T. M. Cover and J. A. Thomas. *Elements of Information Theory*. John Wiley and Sons, 2006. ISBN 0471241954.

M. Dawande, J. Kalagnanam, P. Keskinocak, R. Ravi, and F.S. Salman. Approximation algorithms for the multiple knapsack problem with assignment restrictions. *Journal of Combinatorial Optimization*, 4:171–186, 2000.

B. Deb, S. Bhatnagar, and B. Nath. Information assurance in sensor networks. *In Proceedings of the Second ACM International Workshop Wireless Sensor networks and Applications*, pages 160–168, 2003.

A. Dekorsy, J. Fliege, and M. Śollner. Optimal distributed routing and power control decomposition for wireless networks. *In Proceedings of the Fiftieth IEEE Global Telecommunications Conference*, pages 4920–4924, 2007.

I. Demirkol, C. Ersoy, and F. Alagoz. Mac protocols for wireless sensor networks: a survey. *IEEE Communications Magazine*, 44(4):115–121, 2006.

W. Dinga, S.S. Iyengara, R. Kannana, and W. Rummler. Energy equivalence routing in wireless sensor networks. *Microprocessors and Microsystems*, 28:467–475, 2004.

Y. Duan, Q. Liu, and X. Xu. Application of reinforcement learning in robot soccer. *Engineering Applications of Artificial Intelligence*, 20(7):936–950, 2007.

J. Elson and D. Estrin. Time synchronization forwireless sensor networks. *Proceedings of the 2001 International Parallel and Distributed Processing Symposium (IPDPS), Workshop on Parallel and Distributed Computing Issues in Wireless Networks and Mobile Computing*, 2001.

Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1): 119–139, 1997.

B. R. Frieden. *Science from Fisher Information: A Unification*. Cambridge University Press, 2004. ISBN 0521009111.

E. Gelenbe and R. Lent. Power-aware ad hoc cognitive packet networks. *Ad Hoc Networks Journal*, 2(3):205–216, 2004.

J. C. Gittins. *Multi-armed Bandit Allocation Indices*. John Wiley & Sons, 1989. ISBN 9780471920595.

C. Guestrin, A. Krause, and A. P. Singh. Near-optimal sensor placements in gaussian processes. *In Proceedings of the Twenty-second International Conference on Machine Learning*, pages 265–272, 2005.

S. Guha and K. Munagala. Approximation algorithms for budgeted learning problems. *In the Proceedings of the Thirty-Nineth ACM Symposium on Theory of Computing*, pages 104–113, 2007.

J. P. Hardwick and Q. F. Stout. Bandit strategies for ethical sequential allocation. *Computing Science and Statistics*, 23:421424, 1991.

W. R. Heinzelman, A. Chandrakasan, and H. Balakrishnan. Energy-efficient communication protocol for wireless microsensor networks. *In Proceedings of the Hawaii International Conference on System Sciences*, page 110, 2000.

I. S. Hwang, K. Roy, H. Balakrishnan, and C. Tomlin. A distributed multiple-target identity management algorithm in sensor networks. *In Proceedings of the Fourty-Third IEEE Conference on Decision and Control*, page 72834, 2004.

C. Intanagonwiwat, R. Govindan, and D. Estrin. Directed diffusion for wireless sensor networking. *IEEE/ACM Transactions on Networking*, 11(1):2–16, 2003.

T. Ishikida and P. Varaiya. Multi-armed bandit problem revisited. *Journal of Optimization Theory and Applications*, 83(1):113–154, 1994.

A. Jain and E. Y. Chang. Adaptive sampling for sensor networks. *In Proceedings of the First Workshop on Data Management for Sensor Networks*, pages 10–16, 2004.

N. R. Jennings. An agent-based approach for building complex software systems. *Communications of the ACM*, 44(4):35–41, 2001.

P. Juang, H. Oki, Y. Wang, M. Martonosi, L. S. Peh, and D. Rubenstein. Energy-efficient computing for wildlife tracking: Design tradeoffs and early experiences with zebranet. *In Proceedings of the Tenth International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 96–107, 2002.

L. P. Kaelbling, M. L. Littman, and A. W. Moore. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 4:237–285, 1996.

A. Kansal and M. B Srivastava. An environmental energy harvesting framework for sensor networks. *In Proceedings of the 2003 International Symposium on Low Power Electronics and Design*, pages 481–486, 2003.

J. Kho. *Decentralised Control of Wireless Sensor Networks*. PhD thesis, University of Southampton, School of Electronics and Computer Science, Southampton UK, 2009.

J. Kho, A. Rogers, and N.R. Jennings. Decentralised adaptive sampling of wireless sensor networks. *ACM Transactions on Sensor Networks*, 5(3):19–53, 2009a.

J. Kho, L. Tran-Thanh, A. Rogers, and N. R. Jennings. An agent-based distributed coordination mechanism for wireless visual sensor nodes using dynamic programming. *The Computer Journal*, page doi:10.1093/comjnl/bxp108, 2009b.

R. Kleinberg. Anytime algorithms for multi-armed bandit problems. *In Proceedings of the Seventeenth annual ACM-SIAM symposium on Discrete algorithm*, pages 928–936, 2006.

R. D. Kleinberg. Nearly tight bounds for the continuum-armed bandit problem. In L. K. Saul, Y. Weiss, and L. Bottou, editors, *Advances in Neural Information Processing Systems 17*, pages 697–704. MIT Press, 2005.

R. Kohli, R. Krishnamurti, and P. Mirchandani. Average performance of greedy heuristics for the integer knapsack problem. *European Journal of Operational Research*, 154(1):36–45, 2004.

D.E. Koulouriotis and A. Xanthopoulos. Reinforcement learning and evolutionary algorithms for non-stationary multi-armed bandit problems. *Applied Mathematics and Computation*, 196(2):913–922, 2008.

A. Krause, C. Guestrin, A. Gupta, and J. Kleinberg. Near-optimal sensor placements:maximizing information while minimizing communication cost. *In Proceedings of the Fifth International Conference on Information Processing in Sensor Networks*, pages 2–10, 2006.

T. L. Lai and T. W. Lim. Optimal stopping for brownian motion with applications to sequential analysis and option pricing. *Journal of Statistical Planning and Inference*, 130(1-2):21–47, 2005.

T. L. Lai and H. Robbins. Asymptotically efficient adaptive allocation rules. *Advances in Applied Mathematics*, 6:4–22, 1985.

V. Lesser, C. Ortiz, and M. Tambe, editors. *Distributed sensor networks: a multiagent perspective*. Kluwer Publishing, 2003. ISBN 1402074999.

P. Li, Y. Gu, and B. Zhao. A global-energy-balancing real-time routing in wireless sensor networks. *In Proceedings of the IEEE Asia-Pacific Services Computing Conference*, pages 89–93, 2007.

S. Lindsey and C. S. Raghavendra. Pegasis: Power efficient gathering in sensor information systems. *In Proceedings of the IEEE Aerospace Conference*, 3:3112531130, 2002.

N. Littlestone and M. K. Warmuth. The weighted majority algorithm. *Information and Computation*, 108:212–261, 1994.

M. L. Littman. *Memoryless policies: Theoretical limitations and practical results*. The MIT Press, 1994.

Z. Liu and I. Elhanany. Rl-mac: a reinforcement learning based mac protocol for wireless sensor networks. *International Journal of Sensor Networks*, 1(3-4):117–124, 2006.

O. Madani, D. J. Lizotte, and R. Greiner. The budgeted multi-armed bandit problem. *In the Proceedings of the Seventeenth Annual Conference on Learning Theory, (COLT)*, pages 643–645, 2004.

G. Mainland, D. C. Parkes, and M.Welsh. Decentralised, adaptive resource allocation for sensor networks. *In Proceedings of the Second USENIX/ACM Symposium on Networked Systems Design and Implementation*, pages 315–328, 2005.

A. Mainwaring, D. Culler, J. Polastre, R. Szewczyk, and J. Anderson. Wireless sensor networks for habitat monitoring. *In Proceedings of the First ACM International Workshop on Wireless Sensor Networks and Applications*, pages 88–97, 2002.

K. Martinez, J. Hart, and R. Ong. Environmental sensor networks. *Computer*, 37(8): 50–56, 2004.

G. Mathur, P. Desnoyers, D. Ganesan, and P. Shenoy. Ultra-low power data storage for sensor networks. *Proceedings of the Fifth international conference on Information processing in sensor networks*, pages 374–381, 2006.

G. V. Merrett. *Energy- and Information-Managed Wireless Sensor Networks: Modelling and Simulation*. PhD thesis, University of Southampton, School of Electronics and Computer Science, Southampton UK, 2008.

C. Ok, S. Lee, P. Mitra, and S. Kumara. Distributed energy balanced routing for wireless sensor networks. *Computers & Industrial Engineering*, 57:125–135, 2009.

M. A. Osborne, A. Rogers, S. Ramchurn, S. J. Roberts, and N. R. Jennings. Towards real-time information processing of sensor network data using computationally efficient multi-output gaussian processes. *In Proceedings of the Seventh International Conference on Information Processing in Sensor Networks*, pages 109–120, 2008.

P. Padhy, R. K. Dash, K. Martinez, and N. R. Jennings. A utility-based sensing and communication model for a glacial sensor network. *In Proceedings of the Fifth International Conference on Autonomous Agents and Multiagent Systems*, pages 1353–1360, 2006.

J. B. Predd, S. R. Kulkarni, and H. V. Poor. Distributed learning in wireless sensor networks. *IEEE Signal Processing Magazine*, 23(4):56– 69, 2006.

F. Radlinski, R. Kleinberg, and T. Joachims. Learning diverse rankings with multi-armed bandits. *In the Proceedings of the Twenty-Fifth International Conference on Machine Learning*, pages 784–791, 2008.

V. Raghunathan, C. Schurgers, S. Park, and M.B. Srivastava. Energy-aware wireless microsensor networks. *Signal Processing Magazine*, 19:40–50, 2002.

H. Robbins. Some aspects of the sequential design of experiments. *Bulletin American Mathematical Society*, 55:527–535, 1952.

A. Rogers, D. D. Corkill, and N. R. Jennings. Agent technologies for sensor networks. *IEEE Intelligent Systems*, 24(2):13–17, 2009.

A. Rogers, E. David, and N. R. Jennings. Self-organised routing for wireless microsensor networks. *IEEE Transactions on Systems, Man, and Cybernetics (Part A)*, 35(3):349–359, 2005.

K. Romer and F. Mattern. The design space of wireless sensor networks. *IEEE Wireless Communications*, 11(6):54–61, 2004.

S. Roundy, D. Steingart, L. Frechette P. Wright, and J. Rabaey. Power sources for wireless sensor networks. *Proceedings of the First European workshop on wireless sensor networks*, 2920:1–17, 2004.

S. Schaal and C. G. Atkeson. Robot juggling: An implementation of memory-based learning. *Control Systems Magazine*, 14(1):57–71, 1994.

M. W. M. Seah, C.-K. Tham, V. Srinivasan, and A. Xin. Achieving coverage through distributed reinforcement learning in wireless sensor networks. *In Proceedings of the Third International Conference on Intelligent Sensors, Sensor Networks and Information*, pages 425–430, 2007.

S. Seuken and S. Zilberstein. Formal models and algorithms for decentralized decision making under uncertainty. *Journal of Autonomous Agents and Multi-Agent Systems (JAAMAS)*, 17(2):190–250, 2008.

K. Shah and M. Kumar. Distributed independent reinforcement learning (dirl) approach to resource management in wireless sensor networks. *In Proceedings of the Internatonal Conference on Mobile Adhoc and Sensor Systems*, pages 1–9, 2007.

G. Simon, A. Ledeczi, and M. Maroti. Sensor network-based countersniper system. *In Proceedings of the Second International Conference on Embedded Networked Sensor Systems*, pages 1–12, 2004.

S. Singh, M. Woo, and C. S. Raghavendra. Power-aware routing in mobile ad hoc networks. *In Proceedings of the Fourth annual ACM/IEEE International Conference on Mobile Computing and Networking*, pages 181–190, 1998.

J. A. Stankovic. Research challenges for wireless sensor networks. *ACM SIGBED Review*, 1(2):9–12, 2004.

B. Sundararaman, U. Buy, and A. D. Kshemkalyani. Clock synchronization for wireless sensor networks: a survey. *Ad Hoc Networks*, 3(3):281–323, 2005.

R. S. Sutton and A. G. Barto, editors. *Reinforcement Learning: An Introduction*. MIT Press, 1998. ISBN 0-262-19398-1.

R. Torah, P. Glynne-Jones, M. Tudor, T. O'Donnell, S. Roy, and S. Beeby. Self-powered autonomous wireless sensor node using vibration energy harvesting. *Measurement Science and Technology*, pages 125202.1–125202.8, 2008.

C. Vigorito, D. Ganesan, and A. Barto. Adaptive control of duty-cycling in energy-harvesting wireless sensor networks. *In Proceedings of the Fourth Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks*, pages 21–30, 2007.

D. Wagner and R. Wattenhofer. *Algorithms for Sensor and Ad Hoc Networks: Advanced Lectures*. Springer, 2007. ISBN : 354074990X.

X. Wang and Y. Wang. Optimal investment and consumption with stochastic dividends. *Applied Stochastic Models in Business and Industry*, page doi:10.1002/asmb.823, 2009.

R. Willett, A. Martin, and R. Nowak. Backcasting: Adaptive sampling for sensor networks. *In Proceedings of the Third International Symposium on Information Processing in Sensor Networks*, pages 124–156, 2004.

T. Wu and S. Biswas. Minimizing inter-cluster interference by self-reorganizing mac allocation in sensor networks. *Wireless Networks*, 13(5):691–703, 2007.

J. Y. Yu and S. Mannor. Piecewise-stationary bandit problems with side observations. *In Proceedings of the Twenty-Sixth Annual International Conference on Machine Learning*, pages 1177–1184, 2009.

P. Zhang, C.M. Sadler, S.A. Lyon, and M. Martonosi. Hardware design experiences in zebranet. *In Proceedings of the Second international conference on Embedded networked sensor systems*, 19:227–238, 2004.

Y. Zhang and Q. Huang. A learning-based adaptive routing tree for wireless sensor networks. *Journal of Communications*, 1(2):12–21, 2006.

F. Zhao and L. J. Guibasn. *Wireless Sensor Networks: An Information Processing Approach*. Morgan Kaufmann, 2004. ISBN 1558609148.

J. Zhou and D. de Roure. Floodnet: Coupling adaptive sampling with energy aware routing in a flood warning system. *Journal of Computer Science and Technology*, 22(1):121130, 2007.