COMPUTATIONAL MECHANISM DESIGN FOR WIRELESS SENSOR NETWORKS

Mihail Mihaylov

Master Thesis MICC 08-03

Thesis submitted in partial fulfilment
of the requirements for the degree of
Master of Science in Artificial Intelligence
at the Faculty of Humanities and Sciences
of Maastricht University

Thesis committee:

Dr. K.P. Tuyls
Dr. ir. N. Roos
Dr. ir. R. Peeters
S. de Jong, M.Sc.

Maastricht University
Maastricht ICT Competence Centre
Maastricht, The Netherlands
March 2008

Acknowledgments

I would like to express my sincere gratitude to my supervisor, Dr. Karl Tuyls, for his extensive guidance and invaluable support throughout this thesis, for constantly reminding me not to be *too* worried about small details and for making sure I stay focused on my research. His vast knowledge and expertise in the field of Artificial Intelligence are enviable. I look forward to a continuing collaboration with him in the future.

I would also like to thank my mother, Valentina Mihaylova, for the motivation and support she provided me during my work on this thesis and throughout my life. Additionally, I would like to express my deep appreciation to my brother, Teodor Mihaylov, who, together with my mother, helped me move and settle in a new country – Belgium, and made sure that during their stay there was enough chaos around me to compensate for the monotony of my daily life.

I am grateful also to Prof. Ann Nowé, who gave me the opportunity to continue my education as a Ph.D. student at the Vrije Universiteit Brussel.

Deep acknowledgments go to my bulgarian friends as well – Bozho, Kris, Niki, Stan, Stoycho, Tosho, Veni, Vlado, Yavor. Thank you for your moral support and for making sure I do not stay *too* focused on my research.

Special thanks go also to Kevin for helping me numerous times during my master and for introducing me to the Dutch community and way of life.

I love you all!

Mihail Mihaylov
Maastricht, 12 March 2008

Summary

In this thesis we provide a general overview of Computational Mechanism Design and study its practical application to the energy conservation problem in Wireless Sensor Networks. Mechanism Design is an approach to Game Theory that studies how to make agents achieve the designer's goal out of their own self-interest. Our approach to this problem is based on the Collective Intelligence framework of Wolpert et al, which we regard as Learnable Mechanism Design. COllective INtelligence (COIN) describes how selfish agents can learn to optimize their own performance, so that the performance of the global system is increased.

In our research we study the application of the COIN framework to Wireless Sensor Networks (WSNs), which are collections of densely deployed sensor nodes that gather environmental data. The main challenges in WSN design are the limited power supply of nodes and the need for decentralized control. Therefore, our aim is to increase the autonomous lifetime of the network in a decentralized manner, by making each sensor node use a learning function to optimize its own energy efficiency, so that the energy efficiency of the global system is increased. We show that nodes in WSNs are able develop an energy saving behaviour on their own, when using the COIN framework. We study the performance of different learning algorithms in a simulation environment and provide guidelines on the choice of algorithm for energy efficiency optimization in WSNs for different domains.

Contents

List of Figures

List of Tables

Part I

Theoretical Overview

Chapter 1

Introduction

The aim of this research is to provide a general overview of Computational Mechanism Design and to study its practical application to the energy conservation problem in Wireless Sensor Networks.

Chapter contents: Problem statement, Contributions, Outline.

1.1 Overview

In the first part of this research we examine the field of Mechanism Design and its roots in Game Theory. Mechanism Design (MD) focuses on the implementation of methods that enable a system to reach a desired outcome when self-interested agents with private information interact. MD models the beliefs and private information of agents, and then designs the decision-making protocols (or mechanisms) so that the agents achieve the desired system outcome out of their own self-interest. An example of a mechanism is a traffic light system at an intersection, where impatient drivers (self-interested agents) have to obey the traffic lights (or follow the protocol) in order to reach safely and quickly their destinations (i.e. achieve the designer's goal out of their own self-interest).

We study also the emerging field of COllective INtelligence (COIN), where selfish agents learn to optimize their own performance, so that the performance of the global system is increased. We compare the idea of COIN to MD, because the essence of both approaches is the same, i.e. to make agents achieve the designer's goal out of their own self-interest. Due to this similarity to MD and the fact that agents *learn* to optimize their performance, COIN is often referred to as Learnable Mechanism Design.

In the second part of this research we apply the COIN framework to Wireless Sensor Networks (WSNs). WSN is a collection of densely deployed autonomous devices (nodes) that monitor a large environment with the help of sensors. Sensor nodes have limited power supply, which constitutes one of the major bottlenecks of WSNs. Therefore, using the COIN framework, our aim is to increase the autonomous lifetime of the network, by letting each sensor node optimize its own energy efficiency, so that the energy efficiency of the global system is increased, given the latency requirements of the observer.

1.2 Problem Statement

The main objective of this research is to test experimentally (in a simulator) to what extent Learnable Mechanism Design can be applied to Wireless Sensor Networks in order to extend the autonomous lifetime of the network by improving energy efficiency of nodes. Since the network has to remain functional, our secondary objective is to decrease the latency, caused by energy conservation, so that nodes deliver their messages "in time". Therefore, our problem statement is formulated as follows:

> "To what extent can Learnable Mechanism Design be applied to Wireless Sensor Networks in order to extend the autonomous lifetime of the network?"

The following research questions will guide our work throughout this thesis.

Part I:

- What is the current state of the art in Mechanism Design?
- What is the relation between Mechanism Design and Game Theory?
- What is the relation between Mechanism Design and Collective Intelligence?

Part II:

- What is the current state of the art in Wireless Sensor Networks?
- To what extent can Wireless Sensor Networks develop energy saving schemes when using the COIN framework in combination with learning techniques?

We will investigate the answers to these questions in the next chapters.

1.3 Contributions of the Thesis

The contributions of this research are summarized as follows:

- We provide a theoretical overview of Game Theory, Mechanism Design and Collective Intelligence and survey the state of the art in these fields.
- The relation between Game Theory, Mechanism Design and Collective Intelligence is investigated. We study the roots of Mechanism Design in Game Theory and relate the framework of Collective Intelligence to that of Mechanism Design.
- A simulation tool has been developed for Wireless Sensor Networks, to which the COIN framework has been applied. This simulation tool allows one to compare the performance of different learning algorithms on the energy efficiency and latency of the system.

- We evaluate the performance of different algorithms for energy efficiency optimization and present experimental results that allow the designer to choose different trade-offs between energy efficiency and latency in accordance with his or her requirements.
- We provide guidelines on the choice of algorithm for performance optimization in WSNs for different domains.
- A number of possibilities for future research in different fields is presented.

1.4 Thesis Outline

This thesis is organized as follows: Part I presents the theoretical overview of Game Theory (Chapter 2), Mechanism Design (Chapter 3) and Collective Intelligence (Chapter 4). Part II examines the practical application of the COIN framework on Wireless Sensor Networks. Chapter 5 surveys the state of the art in Wireless Sensor Networks, along with the challenges and opportunities in this field. Chapter 6 studies the theory of using the COIN solution method in WSNs and the difficulties of its application, followed by the practical application of this solution method in Chapter 7. Chapter 8 shows experimental results from different learning algorithms and illustrates the trade-off between energy efficiency and latency. Finally, we present our conclusions and show the opportunities for future research in Chapter 9.

Chapter 2

Game Theory

Game Theory is an economical theory that models the strategic interactions between two or more players, participating in a game.

Chapter contents: Basics, Equilibrium Concepts, Evolutionary Game Theory.

2.1 Introduction

The **game** is the fundamental mathematical construct of **Game Theory** (GT) that involves a given population of individuals, called **agents**, who choose their **actions** from a specific strategy set, according to their utility, and receive **payoffs**, based on the combined action set of all agents participating in the game. A **strategy** is a probability distribution over the available actions. The **utility** of an agent specifies the preference over its actions based on the other players' preferences. The goal of each player is to maximize its payoff by executing actions from its strategy set. In that process, the agent takes into consideration not only its own choices, but also the actions of other participants. The relation between the utilities of the agents for all actions is expressed in a **payoff matrix**.

GT makes two fundamental assumptions [1], when considering agents:

1. Agents are assumed to be rational, i.e. players select actions in order to maximize their payoff;

2. Agents are assumed to be intelligent, i.e., players are able to reason, based on their knowledge in order to select their best response strategy.

To emphasize the strategic aspects of player interaction in non-cooperative games, GT defines two alternative specifications of a game, namely Normal Form Game and Extensive Form Game. The main difference between both is that in the former, all players select their actions at the same time, while in the latter, the agents move consecutively. Extensive Form Games are not of interest for the current research, therefore, we do not discuss them further in this thesis. The interested reader is referred to [2]. The **strategic (or normal) form** representation consists of set of agents, a set of strategies for each agent and a payoff function. This type of representation becomes more convenient when one models the strategic aspects of the agents' behaviour in terms of identifying strictly dominated strategies and Nash equilibria.

In the next section we continue with the definition of a strategy and its sub concepts. In Section 2.3 we mention some static equilibrium concepts of Game Theory and then continue to show the dynamic aspects of player interaction in Section 2.4.

2.2 Strategies

In GT, a **strategy** is a probability distribution over the action set of agents. The set of all players' strategies forms a **strategy profile**. Depending on whether the individual's behaviour is deterministic or stochastic, its action set forms a pure or a mixed strategy, respectively. A **pure strategy** is one which prescribes the choice of a specific action in a deterministic fashion – with a probability of 1. A **mixed strategy** is a probability distribution over actions – the player makes its choices in

a stochastic manner. One can think of pure strategies as a special case of mixed strategies – where all probability is assigned to one strategy.

To illustrate the idea behind the two strategies, we will now show a classical example, often used in the biological literature: the **Hawk-Dove game**. Imagine a large population of a given species, whose individuals are competing for some scarce indivisible resource. The individuals are randomly matched in pairs to display one of the two possible strategies. The first behaviour, known as "*Hawk*", is aggressive – the player always engages the opponent in a fight in order to win the resource, while the second behaviour is more peaceful – the player never fights and will withdraw, if attacked. This is known as the "*Dove*" strategy. The payoffs for both strategies will be discussed later in this chapter.

If an individual always uses one of the two strategies, regardless of the opponent it meets, we say that it is playing a pure strategy. On the other hand, if an individual shows one of the two behaviours with a certain probability (e.g. playing "*Hawk*" 60% of the time and "*Dove*" – 40% of the time), then it uses a mixed strategy. As we shall see later in this chapter, playing only pure strategies is not a profitable decision in the Hawk-Dove game.

A strategy is said to **strictly dominate** another strategy, if the former yields higher payoff than the latter, no matter which actions are played in the latter. If a weak inequality governs the relation between the payoffs of the two action plans, we say that the first **weakly dominates** the second. In a similar fashion, we say that one strategy is **strongly** or **weakly dominated** by another, if the first gives strictly lower, or lower or equal payoff, respectively. A **strongly dominant** strategy is a strategy that produces better outcome than all others, irrelevant of what the other players' strategies are. Similarly, a **weakly dominant**strategy produces an outcome, better than at least one and no worse than all others. In contrast to a dominant strategy, a

best response strategy takes into consideration the other players' strategies in order to maximize the agent's payoff based on its belief what strategy the other players will choose.

The concept of dominance can be illustrated with another classical example of a two-player game – the **Prisoners' Dilemma**, a game introduced by M. Flood [3] and M. Dresher [4] and formalized by A. Tucker (1950). Two suspects of a crime are separately interrogated by the police and offered freedom in exchange to testifying against the other suspect (playing the strategy "*defect*"). If both suspects "*cooperate*" with each other and stay silent, they will receive a lower sentence. More specifically, the payoffs (representing the *negative* of prison years) of the different strategies are given in Table 2.1.

	cooperate	**defect**
cooperate	$(-1, -1)$	$(-5, 0)$
defect	$(\ 0, -5)$	$(-3, -3)$

Table 2.1: Prisoners' Dilemma, payoff matrix

The first value in each cell is the payoff that the first player receives for the joint action, while the second value is the revenue of the second player. It is clear to see that the strategy "*cooperate*" is a strongly dominated strategy, because it produces a strictly lower payoff than the strategy "*defect*", regardless of the behaviour of the other player. Therefore, the strategy defect is the strongly dominant strategy for both players. This example will be further developed in the next subsection to illustrate the concepts of equilibrium.

2.3 Equilibrium Concepts

2.3.1 Nash Equilibrium

Following the assumption that agents are rational, one can expect that players will select the action that maximizes their individual payoff, based on the beliefs about the other players' actions. When each agent selects a strategy that maximizes their utility, one of several equilibrium concepts might arise. A strategy profile is said to constitute a **Nash equilibrium** if no player can improve the revenue of their actions by playing another strategy, while the others keep their strategies unchanged. In other words, a player's strategy is a best response to the other players' strategies, therefore, unilateral deviation from the Nash equilibrium is not rational [5].

When the equilibrium strategy of each agent is pure, we speak of **pure strategy Nash equilibrium**. Since not every game allows players to reach equilibrium only in pure strategies, the pure strategy Nash equilibrium does not exist in every game. On the other hand, if agents use only mixed equilibrium strategies, the strategy profile is called a **mixed strategy Nash equilibrium**. John Nash proved that every n-player finite game has a (not necessarily unique) Nash equilibrium in (possibly) mixed strategies [6].

As we saw in the Prisoners' Dilemma above, the strategy "*defect*" is the dominant strategy for both players. Therefore, it constitutes the single pure strategy Nash Equilibrium, since both players are using only pure strategies and they cannot receive a better payoff by unilaterally changing their behaviour.

However, equilibrium in pure strategies does not always exist. Consider, for example, the **Matching Pennies game**. In this game, two children independently choose to show each other one side of a coin (heads or tails). If both coins show the same side, then the first child wins, otherwise – the second child wins. The payoff matrix of this

game is shown in Table 2.2.

	heads	**tails**
heads	$(+1, -1)$	$(-1, +1)$
tails	$(-1, +1)$	$(+1, -1)$

Table 2.2: Matching Pennies game, payoff matrix

As we know, the first value in each cell is the payoff that the first player receives for the joint action, while the second value is the revenue of the second player. It is clear to see that for each of the four possible pure strategy profiles, one player can always increase its payoff by unilaterally changing its strategy. Therefore, no pure strategy Nash equilibrium exists. However, if players decide to choose one of the two behaviours with equal probability, this would make the other player indifferent to choosing its strategy, therefore, no player has an incentive to deviate. Hence, it follows that the strategy profile, where both players choose their behaviour with equal probability, constitutes the unique mixed strategy Nash equilibrium of the game.

The Nash solution concept is fundamental to game theory [6]. However, it also makes some strong assumptions about the agent's information and beliefs about the knowledge of other agents. For example, Nash equilibrium in a one-shot game requires that the preferences of the agents and the rationality assumption are common knowledge among participants [7]. Furthermore, in case of multiple equilibria, all players must select the same Nash equilibrium. An important impact of this concept is that one can often predict reasonably accurately how agents will behave, if it is assumed they select equilibrium strategies.

When more than one such equilibrium exists, there is no unique solution as to which strategy the agents should follow. Schelling [8] argued that if for any reason, one Nash equilibrium is prominently distinguishable from all others, players will tend to

suppose that other individuals will consider it as well and therefore, follow it themselves. Equilibrium with such property is referred to as **focal equilibrium**.

We will illustrate this term in the following example. Imagine a married couple who has decided to go out on a date together, but they both want to go to different events. The husband wants to go to a football match, while the wife prefers to visit an opera. This game is known as the **Battle of The Sexes**. The payoff matrix is given in Table 2.3

	football	**opera**
football	(3, 2)	(1, 1)
opera	(0, 0)	(2, 3)

Table 2.3: Battle of the Sexes game, payoff matrix

The first value in each cell is the payoff of the husband (row player), if both players choose the corresponding strategies, and the second value is that of the wife (column player) for the same combination. In this game there exist two pure strategy Nash equilibria, i.e. (*football*, *football*) and (*opera*, *opera*). There exists also one mixed Nash equilibrium, where each individual plays its strategy with probability of $\frac{3}{5}$.

Now imagine that for some reason they need to make a decision individually, without the possibility to coordinate with each other. Which Nash equilibrium will they choose? Obviously, they will select their preferred destination with probability $\frac{3}{5}$, but they will never be sure what their partner will choose. Now let us assume both players know that the national football team plays an important game in the town, the evening of their date. In this situation, it is quite reasonable to suppose that the wife will predict her husband will choose to go to the football game. He, in turn, will have the same reasoning and therefore presumes his partner will choose the football game as well. Thus, introducing the above assumption, one of the Nash equilibria becomes

a focal equilibrium. As mentioned earlier, Schelling proposed that if more than one Nash equilibrium exists, agents might choose the one that focuses their attention.

2.3.2 Pareto Efficiency

A Nash equilibrium, however, does not necessarily imply an optimal outcome for all players. Recall the equilibrium strategy profile for Prisoners' Dilemma – (*defect*, *defect*). Examining the payoff matrix in Table 2.1 one could conclude that this is not the optimal solution. Both players will receive a higher payoff if they cooperated, compared to defection. A combination of actions constitutes a **Pareto optimal** (or **Pareto efficient**) solution of a game, iff there is no other such combination with a strongly higher payoff for at least one player and weakly higher payoffs for all others. In other words, in a Pareto optimal solution no agent can increase its payoff, without decreasing the payoff of another. Intuitively, an allocation of payoffs Pareto dominates another if at least one payoff in the first allocation is strictly higher than in the second, and all others are weakly higher. It is easy to see that the strategy profile (*cooperate*, *cooperate*) is a Pareto optimal solution, because none of the players can receive a higher payoff by changing its strategy, without making the other player worse off. Although this strategy profile is strongly preferred by both agents, it does not constitute Nash equilibrium and therefore the players are likely to deviate from their initially chosen strategies to adopt the Pareto dominated solution – (*defect*, *defect*).

2.3.3 Bayesian-Nash Equilibrium

A different solution concept is **Bayesian-Nash equilibrium**, where a prior probability distribution over each agent's preferred actions is common knowledge. When selecting a utility maximizing strategy, an agent considers the expected-utility maximizing strategy of the other agents, computed from the prior distribution over their preferred actions. In this way, the main difference with Nash equilibrium is that the

agent selects a best response to the *distribution* over strategies of other agents, rather than to the *actual* strategies, as in Nash. Bayesian-Nash is a stronger solution concept than Nash, because it uses less assumptions about the information available to each agent, i.e. agents share a *distribution* over preferred actions, rather than the *actual* preferred actions for each agent, as in Nash. However, Bayesian-Nash suffers similar problems concerning the existence of multiple equilibria, information asymmetries and the common knowledge of strategy spaces, payoff functions and rationality.

2.3.4 Dominant Strategy Equilibrium

In **Dominant strategy equilibrium**, agents select their utility maximizing strategy without considering the other agents' choices. In other words, a player's strategy is a best response to any strategy of the other players. Dominant strategy equilibrium is a stronger concept than Nash or Bayesian-Nash and makes no assumptions about the information available to agents or about the common knowledge of rationality. For example, the equilibrium strategy profile in Prisoners' Dilemma is a dominant strategy equilibrium, because agents do not need to model the behaviour of the other agents in the system and predict what actions they will select.

2.4 Evolutionary Game Theory

So far we have examined the static properties of game theory and its solution concepts. We will now focus on the dynamical properties of player interaction in Subsection 2.4.1 and the evolution of a population, playing different pure strategies in Subsection 2.4.2.

2.4.1 Evolutionary Stable Strategy

In contrast to GT, which focuses mainly on the equilibrium properties of strategies, **Evolutionary Game Theory** (EGT) studies the dynamical aspects of strategic

interaction between individuals. It focuses on games where a large population of individuals repeatedly meets in random pairwise identical confrontations. In this way, the strategies, adopted by players, are tested against each other in a repeated manner until the relative frequency of strategies over time is stabilized [9]. In other words, the behaviour of agents determines the outcome of the game, which is represented by the consecutive one-on-one encounters between individuals. This idea of a behaviour, surviving repeated confrontations, forms the central equilibrium concept in EGT – that of **Evolutionary Stable Strategy** (ESS), a term coined by J. M. Smith and G. R. Price in 1973 [10]. A group of individuals is playing ESS when they cannot be invaded by a small fraction, playing a different strategy, known to the population. The higher the fitness of a strategy, the more offspring players with this behaviour it will produce in the next generation. In other words, any alternative strategy, already familiar to players, which appears by mutation or immigration and has lower reproductive success than the original, will not be able to overrule the original strategy and will eventually disappear. It is important to note that the concept of invasion is defined with respect to other strategies, already *known* to players, i.e. it is always potentially vulnerable to *new* (unpredicted) types of behaviour that might arise [9]. A useful characterization of evolutionary stability is that a strategy is an ESS iff it is a best reply to itself and it is a better reply to all other best replies than these are to themselves [11]. Figure 2.1 illustrates the concept of ESS.

We distinguish two types of ESS – Pure and Mixed. When one strategy, regardless of its frequency, totally outperforms all others, we regard it as **Pure ESS**. Such a strategy is immune to invasion by other known strategies [9]. For example, in the Prisoners' Dilemma, the dominant strategy "*defect*" is also pure ESS, because a population of defectors cannot be invaded by a group of individuals, showing the behaviour "*cooperate*". This is simply because "*defect*" is still a better response to "*cooperate*". Figure 2.2 illustrates the concept of evolutionary stability in Prisoners' Dilemma.

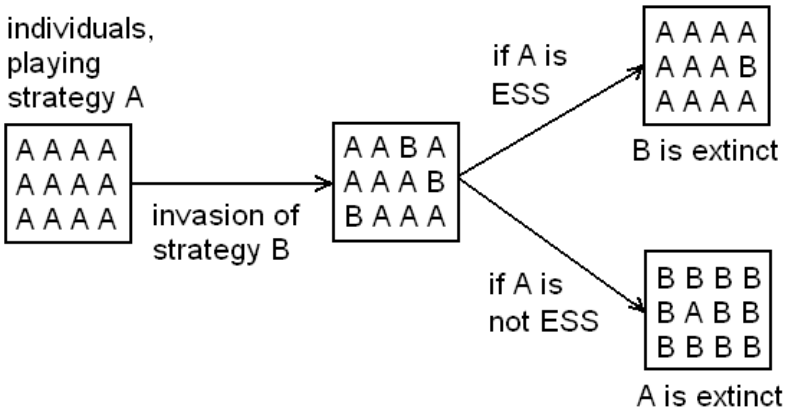

Figure 2.1: Illustration of the ESS concept

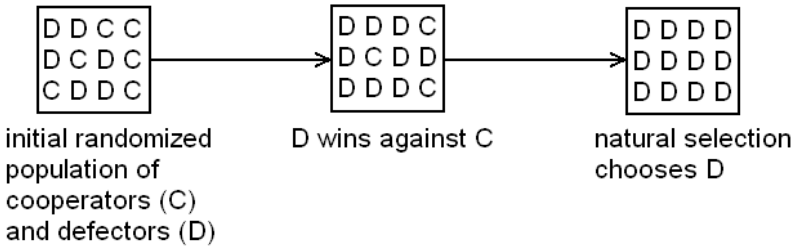

Figure 2.2: Illustration of the evolutionary stability in Prisoners' Dilemma

Where two or more strategies permanently coexist without one being more fit than another, we say the population has adopted a **Mixed ESS**. This can be achieved by playing the strategies at a stable frequency – individuals either play one of the strategies all of the time, or perform the mix of strategies at the equilibrial frequency. In both cases, all individuals will have the same fitness, regardless of the behaviour they adopt. Earlier in this chapter, we briefly described the Hawk-Dove game. We will now elaborate on this game to show examples of ESS.

The payoff matrix for the game of Hawk-Dove is defined in Table 2.4 [12]. where V is the value of the resource and C is the cost that the defeated player has to pay. The first value in each cell is the payoff that the first player receives for the joint action, while the second value is the revenue of the second player. The three possible bilateral confrontations are as follows:

	hawk	**dove**
hawk	$(\frac{V-C}{2}, \frac{V-C}{2})$	$(V, 0)$
dove	$(0, V)$	$(\frac{V}{2}, \frac{V}{2})$

Table 2.4: Hawk-Dove game, payoff matrix

- Hawk vs. Hawk – if both individuals play the aggressive behaviour, they have equal probability (i.e. $\frac{1}{2}$) to obtain the resource, where the defeated player incurs the negative payoff of $-C$, because of the injuries. Negative payoff also results in lower fitness of the Hawk population.

- Dove vs. Dove – if the two individuals adopt the peaceful behaviour, they share the same probability of obtaining the resource, as in the situation above, but this time the losing player does not have to pay any cost. Intuitively, a payoff of 0 has no effect on the fitness of the Dove population.

- Hawk vs. Dove – if both individuals use opposite strategies, the one that shows the Hawk behaviour will obtain the resource with probability 1, whereas the Dove player receives a payoff of 0. In this type of confrontation, Doves always lose the resource to Hawks, but never get hurt.

Examining the payoff matrix, one can confirm that the Dove strategy is not evolutionary stable, because it can be invaded by a small population that has adopted the Hawk behaviour. Depending on the choice of V and C, the following situations might occur (the case where $V = C$ is left as an exercise):

- If $V > C$ then the Hawk strategy will be dominant, because the value of the resource is greater than the cost of injury and therefore it is worth risking the injury to obtain the resource. This is an example of pure ESS, since equilibrium is reached when individuals use only this pure strategy.

- If $V < C$ then no pure Nash equilibrium exists and therefore there is no pure

ESS. In this case, evolutionary stability can be reached when using mixed strategies [2]. Playing the aggressive and peaceful behaviours with certain probabilities results in the unique ESS. This is an example of mixed ESS.

ESS can be regarded as a refinement of the concept of (symmetric) Nash equilibrium [2]. Consequently, the conditions for an ESS are stricter than the ones for NE. Important requirements for evolutionary stability, for example, are that the mutant population is smaller than a certain threshold and that the equilibria, defined by ESS, are symmetric. One implication is that pure strategy NE need not result in an ESS (and in fact, ESS need not exist in a game), while the opposite is true – an ESS is always a NE. Take for instance the game described above. There are two pure NE – (*Hawk*, *Dove*) and (*Dove*, *Hawk*), however, neither *Hawk*, nor *Dove* are ESS. Thus, requirements in evolutionary stability reduce the possible equilibrium points, compared to the Nash concept.

Some important limitations to evolutionary stability [11] are that the concept does not show *how* a population reaches an ESS. It only examines when such a strategy is reached, whether it is stable to invasion. Furthermore, evolutionary stability does not take into account more than one mutant strategy, since it implicitly assumes that mutations happen sufficiently rarely, so that any mutant strategy is displaced, before another one moves in. The ESS concept also does not fruitfully generalize to n-player games, because it then requires that each agent chooses a strategy that is the *unique* best reply to the strategy profile [13].

2.4.2 Replicator Dynamics

The second approach to EGT – **Replicator Dynamics** (RD), due to Taylor and Jonker (1978) [14], examines how a population, playing different pure strategies, evolves over time. The replicator (also gene, organism, etc.) is the central entity in RD that represents one (pure) strategy and has some means of making approximately

accurate copies of itself [15], where all the offspring of a replicator inherit the strategy it represents. During the replication process some random errors, called mutations, might occur, resulting in the appearance of new behaviours. The fitness of the novel strategies will then determine whether those individuals will continue evolving, or go extinct.

To explain the idea behind RD, imagine a very large but finite population of individuals, randomly matched to play a finite symmetric two-player game, using only pure strategies. The fitness of a strategy determines the size of the offspring population, which adopts that behaviour. Intuitively, strategies with higher fitness (or payoffs) reproduce faster. The RD then models the change in the frequency distribution of genes (or pure strategies) over time. In other words, it explicitly models the selection process behind the evolution of a system. The single-population replicator dynamics for symmetric two-player games is given by:

$$\dot{x}_{s'} = [u(x_{s'}, x_s) - u(x_s, x_s)] \cdot x_{s'}$$

where $\dot{x}_{s'}$ is the growth rate of the population share using pure strategy s' and $u(x_{s'}, x_s)$ is the (expected) payoff of these individuals, when playing against the current strategy distribution s of the population. Similarly, $u(x_s, x_s)$ is the payoff of the current strategy against itself. In other words, in a large population, the number of individuals, playing the pure strategy s' will grow exponentially at a rate that equals the difference between that strategy's payoff and the average payoff in the population.

Evolutionary dynamics fruitfully generalizes to n-player games [11] and also provides a dynamical way of reaching equilibrium points for the different behaviours in populations. While every NE is a stable point in RD, the opposite is true only for asymptotically stable equilibria [16].

2.5 Summary

In this chapter we defined the basic concepts of classical Game Theory. We studied the static equilibrium solution concepts and then showed the evolutionary perspective of Game Theory. In the end of this chapter we examined the dynamics of agent interaction.

Chapter 3

Mechanism Design

Mechanism Design focuses on the implementation of methods that enable a system to reach a desired outcome when self-interested agents with private information interact.

Chapter contents: Basics, Properties, Clarke-Groves Mechanisms.

3.1 Roots in Game Theory

There exist two approaches to Game Theory, i.e. Agent Design and Mechanism Design. The previous chapter focused on **Agent Design** (AD). AD approaches the game-theoretic problem from an agent perspective. Game theorists could be seen as external observers who want to know the outcome of a game, but cannot affect this outcome in any way. AD only analyzes agent's decisions and computes the expected utility for each decision in order to find the equilibrium set of strategies for all agents, where deviation is not rational. In other words, AD takes the rules of the game as given and asks the following question:

"Given a game, what is a rational strategy for an agent?" (AD)

The second approach to game theory is called **Mechanism Design** (MD). It focuses on the implementation of methods that enable the system to reach a desired outcome

when self-interested agents with private information interact. In contrast to agent design, MD *defines* the rules of the game and examines the consequences of the different types of rules. Therefore, the basic question that MD addresses is:

"Given that agents are rational, what game should we design?" (MD)

One could easily notice that the question of MD resembles in fact the *inverse* of the question of AD. For this reason, mechanism design is sometimes referred to as **Inverse Game Theory**. It approaches the multi-agent problem in the opposite direction – first modeling the beliefs and private information of agents, and then designing the decision-making protocols (or mechanisms) so that the agents achieve the desired system outcome out of their own self-interest. In this sense, AD can be seen as an approach to analyze the outcome of a mechanism. An important assumption is that agents have private information about their preferences, because if their preferences were public knowledge, the task simplifies to solving an optimization problem, rather than designing a mechanism [17]. In fact, MD could be seen as an optimization problem, to which agents hold the parameters (resembling their private preferences), and the goal is to solve the specified problem, or, in other words, to maximize the value over all agents.

The rest of this chapter is organized as follows. In Section 3.2 we continue with the introduction of the basic concepts in Mechanism Design, followed by the equilibrium concepts in Section 3.3. We mention some desirable properties of mechanisms in Section 3.4 and summarize the different classes of mechanism in Section 3.5.

3.2 Basic Concepts

A key concept in MD is that of a mechanism. A **mechanism** $\mathcal{M} = (S, g(\cdot))$ defines a strategy space $S = S_1 \times \ldots \times S_N$ for agents $N = \{1, 2, \ldots, n\}$, where each agent i has

a strategy space $S_i = \{s_1^i, \ldots, s_{m_i}^i\}$, with $m_i \in \mathcal{N}$ the number of available strategies for agent i; and an **outcome rule** $g : S^N \rightarrow \mathcal{O}$ for some set of outcomes $\mathcal{O}$, that selects outcome $g(s)$ for strategy profile $s = (s_1, \ldots, s_N)$ with $s_i \in S_i$. In other words, a mechanism defines the strategies available to agents and the rules to select the final outcome, i.e. the mechanism specifies how the action profiles are transformed into outcomes. In order to give a more precise definition of a strategy, it becomes useful here to introduce the concept of *type* of an agent. A **type** $\theta_i \in \Theta_i$ for agent $i \in I$, $i = 1, 2, \ldots, n$, determines its preferences over outcomes $o \in \mathcal{O}$. To illustrate this idea, let $u_i(o, \theta_i)$ denote agent i's **utility** function for outcome o, parameterized on the agent's type θ_i. We say that agent i strongly prefers outcome o to o' iff $u_i(o, \theta_i) > u_i(o', \theta_i)$.

It is important to distinguish that the type defines the preferences, which are the *basis* for the utility (i.e. what the agent "generally tends to choose"), while the utility determines the preferences over the *strategies* available to that agent, given its type (i.e. what the agent might choose from the available alternatives). Formally, we have that $u_i : S \times \Theta_i \rightarrow \Re$, i.e. agent i's utility is $u_i(g(s(\theta)), \theta_i)$ (or simply $u_i(s, \theta_i)$) for a strategy profile $s = (s_1, \ldots, s_N)$ and outcome rule $g(s)$, given type θ_i. We know from Chapter 2 that a strategy for an agent selects an action for every distinguishable state of the world. We will now extend the notion of a strategy to include the concept of agent's type. Formally, a strategy $s_i(\theta_i) \in S_i$, with $s_i : \Theta_i \rightarrow S_i$, denotes an action plan for agent i, given type θ_i, in all distinguishable states of the mechanism. The type, therefore, captures all the private information of an agent about its preferences over different outcomes [18].

A mechanism $\mathcal{M} = (S, g(\cdot))$ implements a **social choice function** (SCF) $f : \Theta^N \rightarrow \mathcal{O}$ if:

$$g(s_1^*(\theta_1), \ldots, s_N^*(\theta_N)) = f(\theta), \qquad \forall \theta_i \in \Theta^N$$

where $s^*(\theta)$ denotes an equilibrium strategy profile (irrelevant of the choice of solution concept) to the game $\mathcal{M}$. In other words, the goal of MD is to implement the solution to the SCF in the sense of "game rules", despite agents' self interest. Intuitively, one would prefer to use stronger solution concepts when designing a mechanism (e.g. Dominant strategy equilibrium), because they make less assumptions about agent's information and beliefs about other agents.

Table 3.1 summarizes all basic concepts of MD for an easier reference.

N	Set of agents; $N = \{1, 2, \ldots, n\}$
$\mathcal{M}$	A mechanism; $\mathcal{M} = (S, g(\cdot))$
Θ_i	Set of types of agent i
S	Strategy space; $S = S_1 \times \ldots \times S_N$
$s_i(\cdot)$	Strategy of agent i; $s_i : \Theta_i \rightarrow S_i$
$u_i(\cdot)$	Utility of agent i; $u_i : S \times \Theta_i \rightarrow \Re$
$g(\cdot)$	Outcome rule; $g : S^N \rightarrow \mathcal{O}$
$f(\cdot)$	Social choice function; $f : \Theta^N \rightarrow \mathcal{O}$

Table 3.1: Basic concepts in Mechanism Design

3.3 Equilibrium Concepts

In the previous chapter we explained three solution concepts that are used to compute the outcome of a game in which rational self-interested agents interact (cf. Section 2.3). We will now define those concepts in the setting of mechanism design. For this purpose, we will introduce a notation that will be used in the definition of the equilibrium concepts. We use $s = (s_1, \ldots, s_N)$ to define the strategy profile of all agents and $s_{-i} = (s_1, \ldots, s_{i-1}, s_{i+1}, \ldots, s_N)$ to exclude agent i's strategy from

that profile. In other words, s_{-i} denotes the joint strategies of all agents, except the strategy of agent i. The same holds for θ_{-i}, which stands for the type of every agent, except that of agent i. Furthermore, we use $\hat{\theta}_i$ to denote the type that agent i reports to the mechanism. It need not resemble its true type, because, being rational, the agent will choose to announce a type $\hat{\theta}_i$ that maximizes its expected utility.

We say that a mechanism $\mathcal{M} = (S, g(\cdot))$ implements a SCF $f(\theta)$ in Nash, Bayes-Nash or Dominant strategy equilibrium if

$$g(s^*(\theta)) = f(\theta) \qquad \forall \theta \in \Theta \tag{3.1}$$

where $s^*(\theta)$ is a strategy profile in Nash, Bayes-Nash or Dominant strategy equilibrium, respectively. A strategy profile is in Nash equilibrium, if for every agent i, holds:

$$u_i(s_i^*(\theta_i), s_{-i}^*(\theta_{-i}), \theta_i) \geq u_i(s_i'(\theta_i), s_{-i}^*(\theta_{-i}), \theta_i) \qquad \forall i \in I, \forall \theta_i \in \Theta^N, \forall s_i' \neq s_i^*$$

In other words, every agent i uses strategy s_i that weakly dominates the payoff of any other strategy s_i', given the strategies and types of the other agents. This strategy is then called a **utility-maximizing** strategy. Although it's popularity, Nash equilibrium is very demanding in the assumptions of agents' knowledge, i.e. agents need to be very well-informed about each others' preferences, therefore it is rendered useless in the context of mechanism design [7].

The second equilibrium concept we introduced was Bayes-Nash equilibrium. We will now formalize it to say that a strategy profile is in Bayes-Nash equilibrium if every agent i holds a common prior $F(\theta)$ over the agent types θ, and:

$$E_{\theta_{-i}}[u_i(s_i^*(\theta_i), s_{-i}^*(\theta_{-i}), \theta_i)] \geq E_{\theta_{-i}}[u_i(s_i'(\theta_i), s_{-i}^*(\theta_{-i}), \theta_i)] \qquad \forall i \in I, \forall \theta_i \in \Theta^N, \forall s_i' \neq s_i^*$$

where $E_{\theta_{-i}}[u_i]$ denotes the *expected* utility over the prior $F(\theta)$. As explained in Section 2.3.3, the difference between Nash and Bayes-Nash is that in the latter, agent

i's strategy is a best response to the *distribution* over strategies of the other agents, given the common prior $F(\theta)$, rather than to the *actual* strategies. Therefore, the Bayes-Nash solution concept is more preferred over Nash, since it makes less demanding assumptions about the agents' knowledge and beliefs about other agents.

The third and strongest (or least demanding) solution concept we introduced was Dominant strategy equilibrium. We say that a strategy profile $s^*(\theta)$ is in Dominant strategy equilibrium if, for every agent i, holds:

$$u_i(s_i^*(\theta_i), s_{-i}(\hat{\theta}_{-i}), \theta_i) \geq u_i(s_i'(\theta_i), s_{-i}(\hat{\theta}_{-i}), \theta_i) \qquad \forall i \in I, \forall \theta_i \in \Theta^N, \forall \hat{\theta}_{-i} \in \hat{\Theta}^N, \forall s_i' \neq s_i^*$$

In other words, the equilibrium strategy $s_i^*(\theta_i)$ is a dominant strategy for agent i, if it weakly maximizes its expected utility, whatever the strategies and whatever the types of other agents. We use $s_{-i}(\hat{\theta}_{-i})$ to denote that the agent need not consider the true types of other agents, since its strategy is utility maximizing for all possible strategies and types of other agents. In this way it is clear that the Dominant strategy solution concept makes no assumptions about the knowledge and belief of agents, therefore making it robust and a more preferred solution concept than Nash or Bayes-Nash.

3.4 Properties of Mechanisms

We will now mention several desirable properties of mechanisms. It is useful here to introduce the concept of a **valuation function** $v_i(o, \theta_i)$, parameterized on the type θ_i, such that agent i strongly prefers outcome o to o' iff $v_i(o, \theta_i) > v_i(o', \theta_i)$. We will assume that the valuation functions of all agents are common knowledge.

Section 2.3.2 introduced the concept of Pareto efficiency, which is a preferred property

of a SCF. Formally, a SCF $f(\theta)$ is **Pareto optimal** if:

$$u_i(f(\theta), \theta_i) > u_i(o', \theta_i) \qquad \text{for some } i \in I, o' \neq f(\theta)$$
$$u_i(f(\theta), \theta_i) \geq u_i(o', \theta_i) \qquad \forall i \in I, o' \neq f(\theta)$$

In other words, in a Pareto optimal solution, there is no outcome that is strongly preferred by some agent(s) and weakly preferred by all others. For example, in the Prisoners' Dilemma, the strategy profile (*cooperate*, *cooperate*) is Pareto optimal, because $u_1(cooperate, \theta_1) > u_1(defect, \theta_1)$ for agent 1 and $u_{1,2}(cooperate, \theta_{1,2}) \geq u_{1,2}(defect, \theta_{1,2})$ for both agents 1 and 2. In other words, in strategy profile (*cooperate*, *cooperate*), the strategy *defect* is not strongly preferred by one agent and weakly preferred by both of them, therefore (*cooperate*, *cooperate*) is a Pareto optimal solution.

One drawback of Pareto optimality is that it offers low solution quality, concerning the average agent satisfaction [19]. One could easily demonstrate this by allocating all resources to a single agent and thus obtaining a Pareto optimal solution that is *far* from being preferred by all agents.

A SCF $f(\theta)$ is **allocatively-efficient** if:

$$f(\theta) = \arg\max_{o \in \mathcal{O}} \sum_{i=1}^{N} v_i(o, \theta_i) \qquad \forall \theta \in \Theta^N$$

In other words, an allocatively-efficient SCF selects an outcome that maximizes the sum of agents' valuations, given their types.

A mechanism $\mathcal{M}$ is (*ex-post*) **individually-rational** if it implements a SCF $f(\theta)$, for which:

$$u_i(f(\theta), \theta_i) \geq \bar{u}_i(\theta_i) \qquad \forall \theta \in \Theta^N$$

where $\bar{u}_i(\theta_i)$ denotes the (expected) utility of agent i with type θ_i when *not* participating in the mechanism $\mathcal{M}$. In other words, a mechanism is individually-rational

if participation is voluntary, i.e. if agents may choose whether or not to participate, given their preferences. Moreover, if a rational agent decides to participate, it should never receive a lower payoff than if it did not participate at all, given prior beliefs about the preferences of other agents. For example, the mechanism of the Prisoners' Dilemma is not individually-rational, first because agents are not free to choose whether they want to participate or not (they are prisoners after all) and second because they receive a negative payoff when they participate (cf. Table 2.1). An individually-rational mechanism is an auction, where participation is voluntary and the payoff one receives is always non-negative, provided that he or she did not bid more for the item than it is worth for them. The concept of individual-rationality further decomposes into three types:

- *ex ante* individual-rationality – each agent may choose whether or not to participate, *before* it knows its own preferences. The agent's expected utility when participating in the mechanism, averaged over all possible preferences, must be no less than the expected utility, when not participating. Formally:

$$E_{\theta \in \Theta}[u_i(f(\theta), \theta_i)] \geq E_{\theta_i \in \Theta_i}[\bar{u}_i(\theta_i)] \qquad \forall \theta \in \Theta^N$$

- *interim* individual-rationality – each agent may choose whether or not to participate, when it *knows* its own preferences and has a belief about the *distribution* over the preferences of other agents. Further more, participation should always yield higher or equal payoff, compared to non-participation, given distributional information about the preferences of other agents. Formally:

$$E_{\theta_{-i} \in \Theta_{-i}}[u_i(f(\theta), \theta_i)] \geq \bar{u}_i(\theta_i) \qquad \forall \theta \in \Theta^N$$

- *ex post* individual-rationality – each agent can withdraw from the mechanism when it learns the outcome and that the expected utility from that outcome must not produce a lower payoff compared to no participation, for all preferences of other agents. Formally:

$$u_i(f(\theta), \theta_i) \geq \bar{u}_i(\theta_i) \qquad \forall \theta \in \Theta^N$$

Intuitively, the above three types are presented in ascending difficulty order, i.e. *ex post* individual rationality is most difficult to implement, followed by *interim* and *ex ante*.

A mechanism $\mathcal{M}$ is **incentive-compatible** when the expected utility maximizing (Bayesian-)Nash equilibrium strategy $s_i^* \in S_i$ of every agent $i \in I$ is to report it's true type θ_i out of its own self-interest. Formally:

$$s_i^*(\theta_i) = \theta_i \qquad \forall \theta \in \Theta$$

As an example we will use the famous Vickrey auction [20], in which agents submit sealed bids and the item is sold to the participant with the highest bid at the price of the second highest bid. This mechanism is incentive-compatible, because agents report their true valuations of the auctioned item out of their own interest to obtain it, since lying is not profitable [20].

The concept of incentive-compatibility implies that there is a central mediator that computes a global solution to the reports of agents and then suggests actions, which, when executed by all agents, will achieve the desirable outcome. A drawback to this property is that in an incentive-compatible mechanism, the mediator must assure that the agents will not lie about their private preferences or deviate from the suggested action. This constitutes an enormous computational bottleneck, since the mediators must perform significantly more computation than it would be required for a single agent to solve the whole problem in a centralized fashion [21].

When each agent chooses to announce its true type out of it's own self-interest and this is a dominant strategy for each agent, then the corresponding mechanism $\mathcal{M}$ is **strategy-proof**. This concept can also be defined in terms of incentive-compatibility to say that a mechanism is strategy-proof when incentive-compatibility is implemented in dominant strategies. The Vickrey auction, described above, is an example

of a strategy-proof mechanism, since truth-telling (i.e. bidding one's true value of the item) is a dominant strategy for each agent [20, 7].

Unfortunately, a strategy-proof solution often conflicts with other desirable properties, such as Pareto optimality [22]. Despite its positive features, strategy-proofness is vulnerable to signal noise and computation errors [19]. An unsatisfactory outcome for all agents might emerge from a single (big) mistake, thus making this property sensitive to misstatement.

3.5 Classes of Mechanisms

3.5.1 Direct Revelation Mechanisms

A **direct revelation mechanism** (DRM) $\mathcal{M} = (S, g(\cdot))$ with SCF $f(\theta)$ places restrictions on the strategy set $S = \Theta$ of agents, where the only strategy available to agents is to report a *type* to the mechanism $\mathcal{M}$ with outcome rule $g : \Theta^N \rightarrow \mathcal{O}$, where $g(\theta) = f(\theta)$. In other words, in a DRM, agents will make direct claims about their preferences to the mechanism. Since agents are assumed to be rational, they will choose a strategy $s_i(\theta_i)$ to announce type $\hat{\theta}_i = s_i(\theta_i)$ that maximizes their utility, based on their true type θ_i.

Following the definitions in Section 3.4, if truth-revelation is a (Bayes-)Nash equilibrium in a DRM, then that mechanism is called (Bayes-)Nash **incentive-compatible DRM**, where the outcome $g(\theta) = f(\theta)$ is precisely the SCF implemented by the mechanism. Similarly, if truth-revelation is a dominant strategy equilibrium, the mechanism is called **strategy-proof DRM**. In that case we say that the SCF is truthfully implementable.

An important concept, concerning DRMs, was introduced by A. Gibbard [23], called the **revelation principle**. It states that without loss of generality, any (complex) mechanism can be transformed to an incentive-compatible DRM that implements the same SCF. Formally, if any mechanism $\mathcal{M}$ implements a SCF $f(\cdot)$ in dominant strategies, then $f(\cdot)$ can be implemented in a *strategy-proof* mechanism $\mathcal{M}'$. One implication of this principle is that under quite weak conditions, one could focus on the space of incentive-compatible DRMs[1]. As a result, to identify whether an optimal SCF is implementable in dominant strategies, it is sufficient to show only that it can be *truthfully* implemented in dominant strategies. Another practical implication is that if a SCF can *not* be implemented in a DRM, then it can *not* be implemented in *any* mechanism.

3.5.2 Groves Mechanisms

Groves mechanisms [24] is a family of allocatively-efficient strategy-proof direct-revelation mechanisms. Given the agents' reported types $\hat{\theta} = (\hat{\theta}_1, \ldots, \hat{\theta}_N)$, the mechanism computes the outcome $g(\hat{\theta})$ that solves:

$$g(\hat{\theta}) = \arg\max_{o \in \mathcal{O}} \sum_{i=1}^{N} v_i(o, \hat{\theta}_i)$$

where $\hat{\theta}_i$ is the *reported* preference of agent i, which need not represent its true preference θ_i, due to the rationality assumption (cf. Section 3.3). Recall also that $v_i(o, \hat{\theta}_i)$ represents agent i's valuation for outcome o. Thus, the outcome $g : \Theta^N \to \mathcal{O}$ maximizes the total reported value over all agents. In this way, allocative efficiency follows, because

$$\sum_{i=1}^{N} v_i(g(\hat{\theta}), \hat{\theta}_i) = \max_{o \in \mathcal{O}} \sum_{i=1}^{N} v_i(o, \hat{\theta}_i) \qquad \forall \theta \in \Theta$$

The Groves mechanisms $\mathcal{M} = (\Theta, f(\cdot))$ with SCF $f(\hat{\theta}) = (g(\hat{\theta}), p_1(\hat{\theta}), \ldots, p_n(\hat{\theta}))$ further define payment rule $p_i(g(\hat{\theta}))$, $p_i : \Theta^N \to \Re$, that is associated with the selected

[1] It is usually said that incentive-compatibility "comes for free" [7].

outcome $g(\hat{\theta})$. The payment rule for each agent is defined as follows:

$$p_i(g(\hat{\theta})) = h_i(\hat{\theta}_{-i}) - \sum_{j \neq i} v_j(g(\hat{\theta}), \hat{\theta}_j)$$

for arbitrary function $h_i(\hat{\theta}_{-i})$, $h_i : \Theta_{-i} \rightarrow \Re$, independent of the report of agent i. In fact, the only effect that agent i's report has on its payoff is via the outcome $g(\hat{\theta})$, selected by the mechanism [7]. Therefore, each agent can maximize its utility by announcing its true type, since the outcome itself maximizes the value over all agents. This property determines the strategy-proofness of the Groves mechanisms. Thus, the payment function p_i aligns the agent's goal with the system's goal of an efficient allocation, providing truthfulness in the mechanism.

One limitation to the Groves mechanisms is the computational complexity they show, since the mechanisms provide a centralized solution to a decentralized problem. Furthermore, additional problems complicate the mechanism, such as when agents are not willing to reveal their private information, or simply cannot compute it, or when the centralized optimization problem becomes too complex for the mechanism to compute.

3.5.3 Clarke Mechanisms

The choice of function $h_i(\cdot)$ in the payment rule p_i for agent i defines different subclasses of Groves mechanisms. One such subclass is the **Clarke Mechanisms** [25], which uses the following term:

$$h_i(\hat{\theta}_{-i}) = \arg\max_{o \in \mathcal{O}} \sum_{i \neq j} v_j(g'(\hat{\theta}_{-i}), \hat{\theta}_j)$$

where $g'(\hat{\theta}_{-i})$ is an efficient outcome rule with agent i excluded:

$$g'(\hat{\theta}_{-i}) = \arg\max_{o \in \mathcal{O}} \sum_{i \neq j} v_j(o, \hat{\theta}_j)$$

In this way, agent i's report does not influence its payment and therefore it cannot manipulate its utility by announcing a type, different from its true type. Moreover,

under some quite weak conditions [17], the payment rule $h_i(\cdot)$ in the Clarke mechanism achieves (ex-post) individual rationality while preserving the strategy-proofness and allocative-efficiency of the Groves mechanisms [7]. Using this payment rule, the total payment by agent i to the mechanism represents the effect that the presence of this agent introduces to the other participants. In other words, when agent i is removed from the mechanism, the other agents should be able to achieve at least as much payoff as when agent i is present.

3.6 Summary

In this chapter we studied the Mechanism Design approach and its roots in Game Theory. We defined the basic concepts and explained some equilibrium solution concepts of this field. We showed also some desirable properties of mechanisms and then differentiated the mechanisms in three classes. In the next chapter we will continue with the field of Collective Intelligence.

Chapter 4

Collective Intelligence

The emerging science of Collective Intelligence, by Wolpert et al. [26], considers how to design large multi-agent systems, where selfish agents learn to optimize a private utility function, so that the performance of the global utility is increased.

Chapter contents: Basics, Techniques, Connection to Mechanism Design.

4.1 Introduction

The main difficulty in **COllective INtelligence** (COIN) is that there is little to no centralized control over the behaviour of the computationally bounded agents. Each agent is guided by its own *private utility*, while the performance of the collective is measured by the *world utility*. Thus, the main challenge in the COIN framework is to define suitable private utility functions that will lead to an effective emergent behaviour. The work by Wolpert et at. focuses mainly on the use of reinforcement learning algorithms for optimizing the private utility of agents. The central design problem they consider is:

> "How, without any detailed modeling of the overall system, can one set the utility functions for the RL algorithms in a COIN to have the overall

> dynamics reliably and robustly achieve large values of the provided world utility?" [27]

In the next section we examine the relation between COIN and MD. In Section 4.3 we study the basic concepts of Collective Intelligence and in Section 4.4 we mention some related techniques.

4.2 COIN as Learnable Mechanism Design

Although there are important differences between MD and COIN, their approach is the same, i.e. to make agents achieve the designer's goal out of their own self-interest. In COIN, the designer needs to find a suitable *private utility function*, so that when agents learn to optimize it, they will accomplish the designer's goal. Similarly, in MD he or she needs to find suitable *interaction protocols*, so that agents achieve the intended goal, again because of their own self-interest. Since the basis of the COIN framework is for agents to *learn* to optimize their private utility, COIN is regarded as **Learnable Mechanism Design**. It studies the dynamics of quickly reaching high value solutions (i.e., learning), while classical MD focuses mainly on static environments, by providing incentives that stimulate truthful participation. In other words, there is not yet a dynamic approach in MD that is learnable, but under some assumptions, it is possible to consider COIN as learnable MD. For instance, COIN assumes that agents have enough local information to compute their private utility functions, therefore the COIN framework need not model the private information of agents. Another assumption is that the designer is able to directly set the utility functions of agents, so that the global system behaviour is optimized. The real-world bounded rationality, however, contradicts with the restrictions and assumptions of classical MD and requires run-time adaptive behaviour, which is addressed by the learnable mechanism design approach.

4.3 Basic Concepts

The COIN framework uses a vector ζ that represents the joint actions of all players in the collective, defined in some arbitrary vector space Z. Each agent i acts according to its private utility function $u_i : Z \rightarrow \Re$ that specifies the reward of agent i, given the joint action ζ of the collective. The performance of all agents is then measured by the world utility $G : Z \rightarrow \Re$. The challenge is to find an action vector ζ that aligns the agents' goal of higher individual payoff with the system-wide goal of higher world utility. In other words, the goal is to find suitable private utility functions $u_i(\zeta)$, such that when agents learn to optimize their payoff, this will result in increasing the utility $G(\zeta)$ of the collective system, which is the essence of the learnable MD.

One such private utility function, proposed by Wolpert et al., that is both learnable and aligned with the world utility, is the **Wonderful Life Utility** (WLU):

$$WLU_i(\zeta) = G(\zeta) - G(CL_{S_i^{eff}}(\zeta))$$

where S_i^{eff} is the *effect set* of agent i, i.e. the set of agents that are influenced by the actions of agent i. The function $CL_{S_i^{eff}}(\zeta)$ "clamps" the actions of the effect set of agent i, and thus the function $G(CL_{S_i^{eff}}(\zeta))$ returns the global utility when agent i's effect set is removed from the system. In this way, $WLU_i(\zeta)$ provides the contribution of agent i to the world utility by examining how good the world would perform, had agent i not participated in the system. [1] A crucial strength of WLU is its dynamics-independence, because we do not investigate the evolution of the system when agent i was not present from the beginning [27].

[1] The idea behind this utility function is inspired by Frank Capra's movie *"It's a Wonderful Life"* (1946), where the main character is presented with the fictional idea of what the world would look like, had he never existed.

4.4 Related Techniques

Reinforcement Learning (RL) algorithms [28] are often used to solve problems, addressed by COIN. It examines situations where an unsupervised learner seeks to maximize its payoff by adapting it's behaviour according to the reinforcements it receives. A negative reinforcement expresses a penalty and a positive one indicates a reward. Thus, a selfish agent will try to learn an optimal strategy for the particular problem, by examining the consequences of its actions, without the need to explicitly model the environment. This adaptive characteristic of RL helps to solve the COIN design problem. In particular, COIN seeks to investigate what reward function should each agent adopt, so that when optimized, will result in high world utility.

Another approach, closely resembling COIN is **Swarm Intelligence** (SI) [29], which is a biologically inspired technique, based on the collective behaviour of decentralized, self-organized systems. In particular, SI deals with insect-like behaviour where the decentralized control structure of the system allows for emergent (complex) global behaviour through (simple) local interactions. Thus, social insect colonies are particularly interesting to computer science, since they closely resemble learning in large distributed systems, where "learning" in biological terms is the direct result of natural selection [27]. Despite the similarities with the COIN approach, SI does not offer a general framework for learning to maximize private utilities with the aim to optimize a global value.

4.5 Summary

In this chapter we defined the multi-agent framework of Collective Intelligence, proposed by Wolpert et al and showed why COIN is regarded as Learnable Mechanism Design. We explained some basic concepts of the COIN framework and mentioned

some related techniques. This chapter ends the theoretical part of the thesis. The next part studies the practical application of the COIN framework.

Part II

Application of Mechanism Design to Wireless Sensor Networks: A Case Study

Chapter 5

Wireless Sensor Networks

A Wireless Sensor Network (WSN) is a collection of densely deployed autonomous devices (nodes) that monitor a large environment with the help of sensors.

Chapter contents: Basics, Challenges, Protocols.

5.1 Introduction

The untethered nodes (henceforth also called "sensor nodes", or "sensors") in a Wireless Sensor Network (WSN) use radio communication to transmit sensor measurements to a terminal node, called the sink. The sink is then able to process the distributed measurements and obtain useful information about the monitored environment. WSNs are used in a variety of areas, such as environmental monitoring [30], habitat monitoring [31], in military applications [32], and others. For example, WSNs in habitat monitoring becomes increasingly significant, due to the disturbance effects that human presence introduces to animal populations and plants. The traditional personnel-rich approach, used by researchers in field studies, is usually more expensive and potentially dangerous (e.g. to dormant plants, breeding animals, or even to researchers themselves), as compared to the more economical and less invasive method of wireless sensor monitoring [31]. This remote observation is done by deploying a

set of sensor nodes over the environment of interest at a suitable time (e.g. before breeding seasons) and thus minimizing the human impact on animal populations and plants, by remotely monitoring their habitation.

The typical components of a wireless sensor node are:

- sensor[1], used to make measurements in the environment. The results of this research do not depend on the purpose of the monitoring devices and therefore any sensor can be mounted in the node (e.g. measuring humidity, pressure, temperature, etc.).

- radio transceiver, used to transmit and receive data packages to/from neighbouring nodes. For simplicity, our research uses a single, omnidirectional transmitter, incapable of varying its transmission power; and a single receiver, not able to measure signal strength. The motivation to use such simple devices is to reduce the overall cost of nodes and to keep our solution applicable to the most general sensor network.

- microcontroller, used to control the node's functions. Here, any microcontroller can be used.

- processor, used to partially process the raw measured data in order to reduce the size of the transmitted data.

- battery, used to power the node. The energy requirements of the node constitute a bottleneck for the whole network, due to the limited power supply of each node.

The sensor nodes can vary in size, according to the purpose they serve. They are assumed to be homogeneous and to share a common communication medium (e.g. air, water, etc.). We further assume that the communication range is spherical and

[1]We write the components in singular, but there could be more than one component of the same type (e.g. one sensor for humidity and one for temperature, etc.)

equal in size and strength for all nodes. The omnidirectional antenna therefore can only *broadcast* a message, delivering it to all nodes in range. As explained above, the nodes can neither vary their transmission power in order to "disturb" fewer nodes, nor are they able to estimate their distance from the transmitting node by measuring the signal strength – such features are not generally available in sensor nodes and therefore are not considered here. In order to keep our solution as general as possible, nodes are not assumed to have knowledge of their location as well. To build a sensor network, nodes are randomly scattered in a region. There is a variety of ways of how to spread the sensors, for example by delivering them from a plane or artillery shell, or putting them one by one by a human being or a machine. Depending on the environment they monitor, the nodes can either move (e.g. thrown in water basin) or be static (e.g. mounted in a building[2]). The limited power, processing and communication capabilities of nodes present several challenges to network designers.

The remainder of this chapter is organized as follows. In Section 9.3 we identify some challenges and problems that the wireless sensor monitoring faces. In Section 5.3 we introduce a family of data communication protocols, used in wireless networks. In that section, we identify several sources of energy waste and set the requirements that the WSN communication protocol should meet. We study one protocol in detail and finally present a modified version of that protocol, used in the remainder of this research.

5.2 Challenges

Wireless sensor networks are designed to monitor large and possibly hostile or inaccessible environments. This makes it almost impossible to replace the depleted power sources of large number of nodes in a WSN. The sensors are simply abandoned when

[2]As long as the building stays whole.

their battery is empty. For this reason, one should consider minimizing energy consumption in order to maximize the node's lifetime, while preserving the functionality of the network. Another desirable characteristic of a sensor network is fault tolerance – when one or several nodes run out of battery, the network should be able to dynamically adjust to the new topology and continue its function. The same effect should happen when new nodes are inserted in the network – it should be able to adapt to the new size and/or density of the sensor field. So the three basic design factors that are of primary interest in our research are that a WSN should be:

1. Energy efficient – in order to prolong the lifetime of the network;
2. Scalable – in order to adapt to new nodes;
3. Fault tolerant – in order to adapt to failing nodes.

Initially, nodes in WSNs were used to directly transmit (pre-processed) sensor measurements to a base station, built within nodes' transmission range, which then compiles and further processes the measured data [30]. However, monitoring large environments requires the deployment of high number of devices over (ever increasing) regions, making it difficult to choose a location for the base station that will be in range with all nodes. Increasing the transmission range of nodes, in order to reach the base station, results in a higher interference and energy consumption and therefore decreases the overall lifetime of the sensor field.

To reduce these problems, Zhao et al. [33] proposed a multi-hop communication protocol that allows data packets to be forwarded by neighbouring nodes to the sink, rather than directly transmitting the data to the end point. This solution reduces the requirement for the size of the transmission range and hence, the energy consumption[3], but leads to the necessity of coordination between neighbouring nodes to ensure a viable transmission route. This communication method is called multi-hop

[3]The simplest energy consumption model suggests that the energy, required for transmission, is proportional to the squared distance for this transmission.

routing. It allows for bigger sensor fields, where nodes fall outside the transmission range of the base station. Therefore, a direct centralized control over the network is not possible, so nodes have to organize their schedules and communication in a decentralized fashion. One advantage of the decentralized approach is that different nodes in a sensor field may belong to different stakeholders, serving different purposes, thus making the centralized control a less preferred solution, due to possibly conflicting interests of the stakeholders. A disadvantage of the decentralized approach is that a bad synchronization between nodes may result in lower energy efficiency and/or high latency of the network. The communication protocol, therefore, constitutes an important part of WSN design.

5.3 MAC Protocol

Medium Access Control (MAC) protocol is a data communication protocol, concerned with sharing the wireless transmission medium among the network nodes. Different types of MAC protocols exist, but we will focus mainly on those that take into consideration energy efficiency of the network and the latency requirements of the observer.

The multi-hop routing, described above, is also adopted by some wireless ad-hoc networks for the numerous purposes, explained in Section 5.1. However, the protocols, used by ad-hoc networks cannot be applied to WSNs, due to several differences between the two types of networks [34]. Some differences include the large number and density of sensor nodes, compared to the nodes in ad-hoc networks, the frequently changing topology of sensor nodes and their power and processing constraints, etc. Furthermore, MAC protocols for traditional wireless networks do not take into account the primary design factor of WSNs – energy efficiency, but rather focus on characteristics that are of secondary importance for WSNs, like fairness, throughput and optimal bandwidth utilization. Before we focus on any specific MAC protocol,

we first identify the four different modes in which a sensor node can operate.

5.3.1 Operation Modes

Each node operates according to a schedule that defines four different modes – transmit, listen, receive and sleep. We can specify them as follows:

- a node goes in **transmit** mode, when it starts to send a message through the channel. The omnidirectional antenna of the node broadcasts the message to all nodes in range, which we call *neighbours* (or *neighbouring nodes*). The MAC protocol is only concerned with efficient node-to-node delivery, while the end-to-end delivery is handled by the routing protocol. Therefore, neighbouring nodes should discard a broadcasted message, not addressed to them.

- when in **listen** mode, the sensor node is actively listening for broadcasts in the medium. When a signal is detected, the node goes in receiving mode.

- a sensor node is in **receive** mode when receiving a message from a neighbouring node. When the whole message is received, the node goes in one of the three other modes.

- when a node is in **sleep** mode, its radio transceiver is switched off and therefore no communication is possible. Nevertheless, the node continues its sensing and processing tasks.

These four operation modes pose potential problems to the communication, because two nodes have to be synchronized with each other, prior to exchanging data. Two nodes are synchronized, when the sender is in transmit mode, while the receiver is in listen mode. Only then, a successful transmission can take place, provided no collisions occur at the receiver's antenna. A collision happens when more than one packet arrives at the same time at the node's antenna. There are some types of MAC protocols that handle collisions, but those events are not of primary interest to

this research and therefore such protocols are not considered. Taking into account the energy requirements of the different modes, we identify several sources of energy waste.

5.3.2 Energy Waste

To identify the major sources of energy waste, we have to define the energy consumption rules for the different modes. We present those rules, relative to the different modes, rather than by using exact values, because energy consumption is hardware dependent. The relative energy, required by each mode is as follows:

- transmitting is two times more expensive (in energy consumption) than listening or receiving;
- listening and receiving consume nearly the same energy;
- sleeping is ten times cheaper than listening or receiving.

Given the relative energy consumption rules above, we can identify four major sources of energy waste – idle listening, overhearing, collision and control packet overhead. They are explained as follows:

- **idle listening** happens when a node is listening to the channel when no neighbour is transmitting a message. Since no messages are being sent, the node is better off sleeping, because it is ten times cheaper than listening.
- **overhearing** occurs when a node receives a packed that is addressed to another node and not for itself. This event can happen due to the broadcasting nature of the communication. To avoid this problem, the overhearing node is better off sleeping.
- **collision** is another event that happens at the receiver's antenna, i.e., the node receives more than one package at the same time. In this case energy is wasted,

because the message was not received properly and therefore needs to be retransmitted. This problem can be avoided, if the sender of the message that caused collision was sleeping during the communication of the neighbouring nodes.

- **control packet overhead** represents the energy loss due to the exchange of control packets prior to and during the transmission of the actual message. The frequency of the control packets should be kept low to minimize the effect of this problem.

In order to maximize energy efficiency of the network, our communication protocol should minimize the above sources of energy waste, while maximizing sleep mode and considering the latency requirements of the observer.

5.3.3 The Choice of Protocol

The choice of a suitable MAC protocol is research on its own, therefore we will not analyze and compare different types of protocols here. We will mention just two possible solutions and will motivate our choice of one of them, although that choice need not be the optimal protocol for our research.

Beyens et al. [35] proposed the use of Time Division Multiple Access (TDMA) class of MAC protocols, where the signal is divided into frames and each frame into time slots, which are assigned by the protocol to each node. This allows many nodes to use different parts of the bandwidth of the same radio channel. The schedule of each node determines the amount of time slots it will use for each mode in the current frame, so that neighbouring nodes can synchronize their communication easily and avoid collisions and overhearing. As a result of the time slot allocation, each node is allowed to communicate with at most one neighbouring node at a time. This allocation also allows nodes to share their schedules by broadcasting them, which allows for easier synchronization.

The proposed TDMA protocol in [35] uses only one control packet – the acknowledgment (ACK). This control packet is sent by the receiving node to the sender in order to verify that the last data (DAT) packet was received successfully. In other words, after sending each DAT packet, the sender waits for an ACK from the receiver, in order to make sure the DAT packed was received correctly, before sending the next one. Although using a single control packet has the advantage of low communication overhead, it is not enough to ensure well-synchronized communication, avoidance of collisions and overhearing.

5.3.4 Sensor-MAC Protocol

The Sensor-MAC (S-MAC) protocol, proposed by Wei Ye et al. [36], is a new energy-efficient MAC protocol, explicitly designed for WSNs. It uses additional control packets, similar to those in the IEEE 802.11 standard, that help minimize the major sources of energy waste, explained in Section 5.3.2. Prior to transmitting a DAT packet, the sender broadcasts a Request to Send (RTS) control packet, addressed to the intended receiver in order to reserve the medium for the following transmission and synchronize the communication with the receiving node. If the receiver hears the RTS packet, it responds with a Clear to Send (CTS) control packet, in order to notify the sender that it has reserved its schedule for the following transmission and is ready to receive data.

The S-MAC protocol also reduces the listening time of nodes by making them go into periodic sleep mode. The duration of the periodic sleeping is set in accordance with the latency requirement of the user. For example, if this requirement allows that the nodes sleep half of the frame (and communicate in the other half), then the energy efficiency is increased with nearly 50%. The S-MAC protocol uses collision and overhearing avoidance schemes by putting a duration field in each packet, indi-

cating the duration for which the medium is reserved. In this way, when a packet is overheard by a node, that node will know how much time it has to stay silent in order to avoid collisions at the receiver, or even go to sleep, in order to avoid overhearing the remaining communication.

As a result of the collision and overhearing avoidance schemes, the following scenario happens when two nodes communicate. In the multi-hop network of Figure 5.1, node C broadcasts an RTS packet, intended for node D. The latter, in turn, replies with a CTS packet, addressed to node C. The big circles show the transmission range of the communicating nodes. In this scenario, node E has to go to sleep, so that it avoids overhearing the ACK packets of D or causing collision in D's antenna upon transmission to F. Since node B will overhear the DAT packets of node C and detect collisions by messages, coming from A, node B is better of sleeping. In other words, the immediate neighbours of the communication partners have to sleep during the whole communication in order to avoid overhearing and collisions. It is clear that nodes A and F will not interfere with the communication of nodes C and D, therefore can remain active.

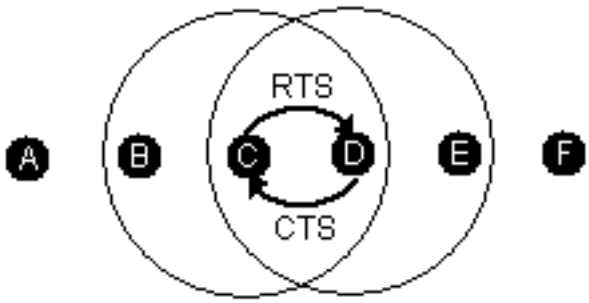

Figure 5.1: Collision and overhearing avoidance during communication

Although TDMA protocols increase energy conservation, they require that nodes form *real* communication clusters, like Bluetooth [37]. Since managing inter-cluster com-

munication is usually a difficult task, this requirement is relaxed in S-MAC, where nodes form a flat topology – sensor nodes are free to communicate with each other, no matter what listen schedules they have [36]. Moreover, in TDMA protocols it is not easy to change the frame length when the WSN changes in size, while S-MAC is well-scalable for a dynamic topology. However, the structure of the S-MAC protocol is not *entirely* suitable for the purpose of our research. This will be explained in the next section, where we introduce a modified version of the proposed protocol.

5.3.5 Sensor-MAC Protocol (Modified)

In order to reach the goals of the current research, we have made several modifications of the S-MAC protocol, presented above. For reasons of simplicity, our version of the protocol does not use virtual or physical carrier sense, as described in [36]. Moreover, the function of the SYN packet (explained below) is modified. As a result, we do not divide the listening interval of a node in two parts, as it is done in the original version of the S-MAC protocol. We summarize the functions of each (control) packet in our protocol as follows:

- Request to Send (RTS) – a control packet, transmitted by the initiator of the communication in order to reserve the medium for the following transmission and therefore to tell the other neighbours to go in sleep mode, since they have to stay silent for the specified duration. Additionally, this packet also informs the receiver and all neighbours about the duration of the communication, so that they can modify their schedules accordingly.

- Clear to Send (CTS) – a control packet, transmitted by the receiver in the communication, as a reply to RTS, in order to acknowledge the communication and inform its neighbours to go in sleep mode for its duration.

- Data (DAT) – a data packet that contains the actual message, transmitted by the sender to the receiver. Each DAT packet has to be acknowledged, so that

the communication can continue.

- Acknowledgment (ACK) – a control packet, transmitted by the receiver in the communication, as a reply to DAT, in order to acknowledge the successful reception of the last data packet.

- Synchronization (SYN) – a control packet, broadcasted by each node during an initial synchronization of the network – a process which determines the number of nodes between each node and the sink. Nodes use this information in order to route a message toward the sink.

To illustrate the idea of collision and overhearing avoidance and the use of (control) packets, we will show an example of communication between nodes C and D from Figure 5.1. This example is shown in Figure 5.2. In this scenario, node C initiates a communication to node D, by sending an RTS packet, which specifies the duration of the whole communication (in time slots)[4]. In the next slot, node D acknowledges this communication by replying with a CTS packet, containing the original duration, decremented by one, because one slot has already passed. The actual transmission of the message happens next between the two communication partners, by exchanging DAT and ACK packets, each time with a decremented duration field. Note that the last ACK packet of node D has a duration field 0, because the "remaining communication" will take no more slots. In this example, node B is still listening from the previous frame and therefore overhears the RTS packet of node C (recall that nodes B and C are neighbours). The duration field of that packet contains the value 5, therefore node B goes in sleep mode for this duration, in order to avoid overhearing the entire communication. Similar is the case with node E, which is a neighbour of node D. In the third slot of the new frame, node E goes in listening mode for the remaining duration of the current frame (8 more slots), when it overhears an ACK

[4]The duration field specifies the amount of time slots that the *remaining* communication will take, *after* dispatching the corresponding packet. For this reason, the duration field in node C's RTS packet is 5, while in fact, the whole communication occupies 6 slots.

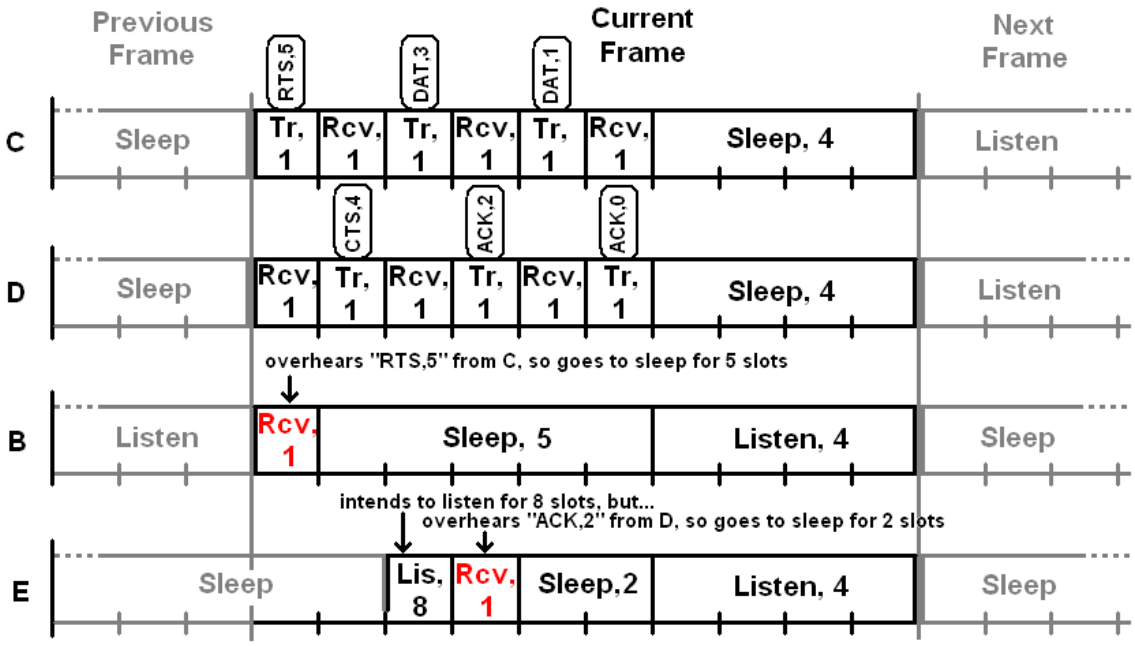

Figure 5.2: Example of communication and overhearing – the use of packets

packet of node D, containing the duration of 2. Node E immediately goes to sleep for 2 slots, until the communication between C and D is over. After the communication has ended, the nodes resume their periodic schedules, i.e. nodes C and D go to sleep for the remaining duration of the frame (4 more slots), because they have been communicating before, and nodes B and E go in listening mode (for the same duration), because they have been sleeping before. The same behaviour continues in the next frame.

5.3.6 Summary

In this chapter we presented the wireless sensor network and its application in remotely monitoring large environments with the help of small untethered sensor nodes. We identified the challenges and limitations that such a network faces, in terms of computation and energy constraints, and proposed a communication protocol that overcomes some of the problems in wireless sensor monitoring.

Chapter 6

COIN as a Solution Method

In this chapter we will apply the COIN framework, discussed in Chapter 4, to the MAC optimization problem, by defining the world utility and the private utility of the agents.

Chapter contents: Computation of the Wonderful Life Utility.

6.1 Introduction

Since our main objective is to increase the energy efficiency of the sensor nodes in a WSN, we first need to define what an energy efficient behaviour is and how it can be achieved. In Section 5.3.2 we argued that an energy efficient behaviour should minimize the major sources of energy waste, i.e. idle listening, overhearing, collision and control packet overhead, while maximizing sleep mode and taking into account the latency requirement of the observer. The latter requirement specifies the maximum amount of time that a message, once created, can be forwarded before reaching the sink (i.e. the observer).

In Section 6.2 we describe how each node can compute its energy efficiency and then integrate it in the computation of the WLU in Section 6.3. In the latter section we

mention some difficulties in obtaining the components of the private utility function and propose methods to overcome these problems.

6.2 Computation of Energy Efficiency

In order to compute its energy efficiency (EE) for the next set of timeslots, i.e., for the next frame, a node i needs to determine the following variables and update them at the beginning of each frame f:

- $\sigma_{i,f}$ is the number of time slots in frame f, during which agent i is sleeping
- $\lambda_{i,f}$ is the number of time slots in frame f, during which agent i is listening
- $\rho_{i,f}$ is the number of time slots in frame f, during which agent i is first listening and then receiving data
- $\theta_{i,f}$ is the number of successful transmissions of data packets of agent i in frame f (ACK received)
- $\nu_{i,f}$ is the number of unsuccessful transmissions of data packets of agent i in frame f (ACK not received)
- $o_{i,f}$ is the number of overheard packets by agent i in frame f
- $packets_{i,f}$ is the number of data packets in the queue of agent i at the beginning of frame f
- $frames_{i,f}$ is the sum of the number of frames that each packet is waiting in the queue of agent i for frame f
- $latency_{max}$ is the maximum number of frames that each packet is allowed to stay in the queue of an agent
- $slots_f$ is the number of time slots in a frame

At the beginning of each frame, every node updates these variable in order to compute its EE for the last frame. We define the energy efficiency of an agent in terms of the following components[1]: idle listening, overhearing, unsuccessful transmissions, sleeping and latency, where each component lies within the interval $[0, 1]$.

The **idle listening component** gives an indication of energy waste by idle listening and is defined as:

$$IL_{i,f} = \frac{\lambda_{i,f}}{\lambda_{i,f} + \rho_{i,f}}$$

The **overhearing component** gives an indication of energy waste by overhearing and is defined as:

$$OH_{i,f} = \frac{o_{i,f}}{\rho_{i,f}}$$

The **unsuccessful transmissions component** gives an indication of energy waste by unsuccessful transmissions, i.e. ACK not received for a DAT packet, and is defined as:

$$UT_{i,f} = \frac{\nu_{i,f}}{\nu_{i,f} + \theta_{i,f}}$$

The **sleeping component** expresses the number of time slots in frame f, during which agent i is sleeping and is defined as:

$$SL_{i,f} = \frac{\sigma_{i,f}}{slots_f}$$

The **latency component** expresses the latency, as specified from the observer, and is defined as:

$$LA_{i,f} = \frac{frames_{i,f}}{packets_{i,f} * latency_{max}}$$

Each nodes computes these components at the beginning of every frame in order to determine its **energy efficiency** for the joint action ζ of all agents. The energy efficiency is defined as:

$$EE_{i,f}(\zeta) = \alpha(1 - IL_{i,f}) + \beta(1 - OH_{i,f}) + \gamma(1 - UT_{i,f}) + \delta(SL_{i,f}) + \epsilon(1 - LA_{i,f})$$

[1]Both the variables and the components are copied from [35] with the permission of Karl Tuyls.

where the parameters α, β, γ, δ and ϵ weight the importance[2] of the different terms in the above formula and they sum up to 1. Thus, the energy efficiency $EE_{i,f}(\zeta) : \zeta \rightarrow \Re$ lies within the interval $[0, 1]$. Note that the components, which indicate energy waste, are subtracted from one in order to express that higher values of $EE_{i,f}(\zeta)$ mean higher energy efficiency of the node. Once we are able to compute the EE of each node, we can proceed with the formulation of the WLU.

6.3 Computation of WLU

In Chapter 4 we defined the Wonderful Life Utility of agent i for the joint action ζ of all agents as follows:

$$WLU_i(\zeta) = G(\zeta) - G(CL_{S_i^{eff}}(\zeta))$$

We now define the world utility $G(\zeta)$ as:

$$G(\zeta) = \frac{1}{F} \sum_f \frac{1}{N} \sum_i EE_{i,f}(\zeta)$$

where F is the number of frames that have passed until now and N is the number of agents in the system. Thus, the world utility is the energy efficiency of all agents over time. To be able to compute its EE *over time*, each node uses a frame window W that stores the set of $|W|$ past frames, over which the EE will be averaged. In this way, the energy efficiency of agent i in the frame window W of size $|W|$ is given by:

$$EE_{i,W}(\zeta) = \frac{1}{|W|} \sum_{f \in W} EE_{i,f}(\zeta)$$

In order to increase the autonomous lifetime of the network, each agent will try to maximize its private utility, i.e. the WLU, so that the world utility is increased as well. However, the nodes face two difficulties in computing the WLU:

[2]One could use the relative energy consumption rules of the different modes in order to set the importance values (cf. Section 5.3.2).

1. Sensor nodes cannot compute the world utility $G(\zeta)$, because they require the EE of *every* agent in the system, which they are not possible to obtain, due to the limited transmission range – sensors can only obtain information that is at most a few hops away.

2. Sensor nodes cannot accurately determine the size of their effect set S_i^{eff}, because the result of each action can influence different amount of nodes and therefore this will lead to a different size of the effected set for each action.

To overcome these difficulties, we will use approximations[3] to compute the world utility and the effect set. For this reason, we will introduce a set $N_{i,h_{min},h_{max}}$ that holds the set of agents, which are at least h_{min} hops away from agent i, but less than h_{max} hops away (h_{min}, $h_{max} \in \mathcal{N}$).[4] The number of agents in the set $N_{i,h_{min},h_{max}}$ is given by $|N_{i,h_{min},h_{max}}|$. Using this set, nodes can approximate the world utility $G(\zeta)$ by considering only those nodes that are less than h_{world} hops away ($h_{world} \in \mathcal{N}$). In this way, the **estimated world utility** $G_i'(\zeta) : \zeta \rightarrow \Re$ for agent i is defined as:

$$G_i'(\zeta) = \frac{1}{|N_{i,0,h_{world}}|} \sum_j EE_{j,W}(\zeta) \qquad j \in N_{i,0,h_{world}}$$

This definition overcomes the first problem, presented above. The second problem, that of defining an effect set, can be overcome by introducing the **guessed effect set** of agent i as:

$$S_i^{eff} = N_{i,0,h_{eff}} \qquad h_{eff} \in \mathcal{N}$$

that holds the set of agents, which are at most h_{eff} hops away from agent i. Figure 6.1 illustrates the two different ranges for approximation – h_{world} and h_{eff}, compared to the transmission range of a node. Using this definition, we can compute the second

[3]Wolpert et al. have shown that approximations still provide good results in a load balancing problem in data networks [38].

[4]When a node is n hops away from another, it means that its message needs to be forwarded at least n times to reach the other node. Therefore, nodes that are one hop away from each other are neighbours, i.e. they lie within each other's transmission range.

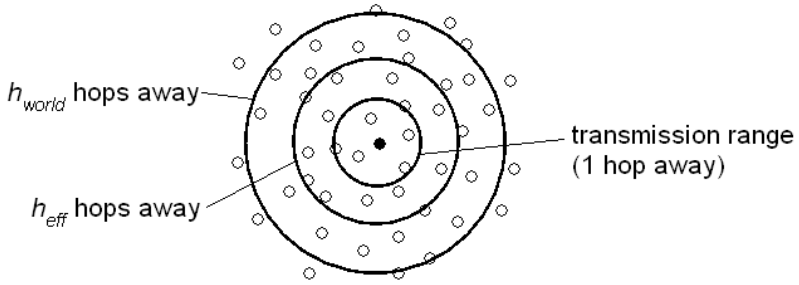

Figure 6.1: Hops for approximating the world utility and the effect set of a node

term in the WLU_i of agent i:

$$G(CL_{S_i^{eff}}(\zeta)) = \frac{1}{|N_{i,h_{eff},h_{world}}|} \sum_j EE_{i,W}(\zeta) \qquad j \in N_{i,h_{eff},h_{world}}$$

This term gives us the contribution of the agents that are not part of the guessed effect set of agent i (cf. Section 4.3). With those three definitions we are now able to compute the $WLU_i(\zeta)$ of agent i for the joint action ζ of all agents as:

$$WLU_i(\zeta) = G'_i(\zeta) - G(CL_{S_i^{eff}}(\zeta))$$

which provides the contribution of agent i to the energy efficiency of the system. Therefore, if an agent learns how to optimize its private utility, the global utility will increase as well, which will lead to a higher energy efficiency and therefore increased autonomous lifetime of the system. In other words, by implementing a learning algorithm, nodes should be able to find an energy efficient solution by themselves and be able to quickly adapt to new or failing nodes, as compared to nodes in a typical ad-hoc network. One drawback of our approach is that nodes require information of neighbouring nodes, which results in an increased overhead per data packet. In the next chapter, we will examine the effects of three algorithms for WLU optimization on the energy efficiency of different WSNs.

6.4 Summary

In this chapter we presented a method for nodes to compute their energy efficiency over time. We used the COIN solution method, described in Chapter 4, to align the private utility of each agent with the global utility of the system. We identified some difficulties of the theoretical approach of COIN in obtaining the components of the wonderful life private utility and proposed a solution how to avoid these problems. We showed how every agent is able to approximate its contribution to the global utility and thus try to improve its performance, so that the energy efficiency of the WSN can be improved as well.

Chapter 7

Practical Application of the COIN Solution Method

In this chapter we will examine the effects of three algorithms that sensor nodes use to optimize their wonderful life private utility with the aim that the global utility, and hence – energy efficiency, increases.

Chapter contents: Learning algorithms, Obtaining the Wonderful Life Utility.

7.1 Introduction

Experimental results will evaluate the performance of each algorithm, compared to the case where the node did not use *any* algorithm for optimization. All three algorithms are designed to influence in different ways a single variable – the **fatigue**. Each node uses this variable to determine its periodic schedules, i.e. the amount of timeslots a node sleeps and listens in a frame. A high value of the fatigue means that the node will listen less timeslots and sleep more (hence the name "fatigue"), while a low value indicates that the node will spend more time listening to the medium and therefore, will participate in communications, and sleep less time. The motivation behind choosing this method of regulating the node's energy efficiency is straightforward – more sleeping means higher energy efficiency of the network, but also higher

latency, because each message will be delivered more slowly to the sink. More listening, on the other hand, will speed up message delivery, i.e. decrease the latency, but will also decrease energy efficiency, due to the increased communication and interference with neighbouring nodes. Figure 7.1 illustrates the influence of fatigue on the durations of the different modes of a node. The sensors need to find a good tradeoff between EE and latency, by using different periodic schedules, so that they combine the benefits of both behaviours (more sleeping vs. more listening).

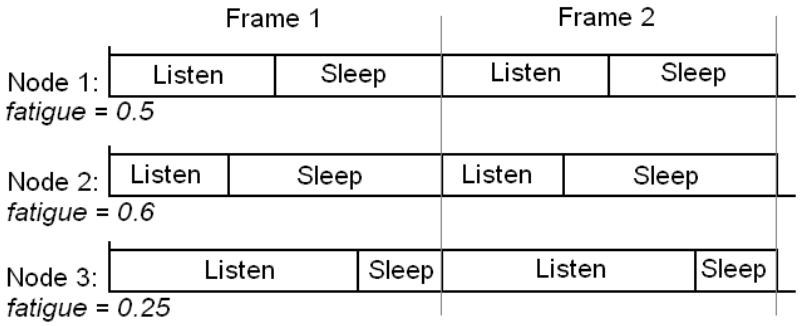

Figure 7.1: Periodic schedules for different values of *fatigue*

The coordination between the behaviour of different nodes in a WSN resembles the **El Farol Bar problem** [39] where a finite population of individuals has to independently decide each day at the same moment whether they will visit this bar or stay at home. If many people choose to visit the bar on the same day, the bar will become overcrowded and everyone will have a worse time being there than staying at home. If, on the other hand, fewer people go to that bar, they will enjoy it more than if they stayed at home. In summary, if everyone uses the same pure strategy (go to bar vs. stay at home), they will not achieve a Nash equilibrium. The agents should, therefore, adopt mixed strategies, i.e. choose a day with a certain probability, in order to be distributed evenly in the bar through the week and in this way achieve higher payoff. Back to the WSN, the problem of choosing the right schedule at the right moment resembles the El Farol Bar problem, because each node should sleep more or listen more, with a certain probability in order to achieve higher energy efficiency of the

network. Using the same pure strategy (sleep more vs. listen more) for every node, will not achieve high energy efficiency *and* low latency of the WSN.

In Section 7.2 we will propose the use of three algorithms for private utility optimization in order to achieve higher world utility *and* lower latency of the network. In Section 7.3 we will explain how a node can obtain the components of the WLU, i.e., the world utility and the contribution of the agents, outside the node's effect set. We will describe the practical implementation of our communication protocol for obtaining those values.

7.2 Algorithms for WLU Optimization

As we argued above, the problem of independently choosing a schedule in each frame resembles the El Farol Bar problem of independently choosing a day to visit the bar. The solution of the latter problem suggests that nodes should adopt mixed strategies in order to achieve higher individual payoff. Translating this solution to our WSN, each node should sleep more or listen more, with a certain probability, in order to achieve higher world utility. Therefore, the different values of *fatigue* will ensure that nodes sleep and listen different amounts of time slots to achieve low latency *and* high energy efficiency of the network. To obtain this result, we investigated the influence of three algorithms on *fatigue*. Each algorithm is "tweaked" by a value, called *param*, which is set *at the beginning of the simulation* and therefore can be seen as a constant, during runtime. The influence of different parameters on the algorithms is examined later in this chapter. The three algorithms are described as follows:

1. The idea behind this algorithm lies in the suggestion that low energy efficiency might be caused by nodes trying to send messages, when their neighbours are sleeping, i.e. by unsuccessful transmissions, and hence – control packet overhead. To fix that, nodes should become more active and therefore exchange

messages in a more efficient way, leading to an increase in WLU. In other words, this rule weights the importance of the energy waste by control packet overhead more than that of idle listening and overhearing. This algorithm is a decision rule, shown in Listing 7.1. The node stores the value of its WLU from the last frame in `old_WLU` and compares it to the `WLU` of the new frame. If the the value has decreased (irrelevant of the magnitude), the rule decrements `fatigue` by the constant value `param`, defined at the beginning of the simulation. If `WLU` has increased, `fatigue` is incremented by twice that constant.[1]

Listing 7.1: Algorithm 1 – a decision rule

```
if (WLU < old_WLU)
    fatigue = fatigue - (param/100);
else if (WLU > old_WLU)
    fatigue = fatigue + (param/100)*2;
```

2. The purpose of this algorithm is to make each node adopt *literally* mixed strategies, i.e. to choose to sleep more or to listen more with a certain probability, so that a good tradeoff between latency and EE is achieved. This algorithm, shown in Listing 7.2, tells the node to *ignore* its WLU at all and change `fatigue` according to a random variable (rather than its private utility), which range is defined by `param` at the beginning of the simulation. This will ensure that the periodic schedules of nodes will be evenly balanced – while some nodes are more active in communication (to ensure low latency), others sleep more (to ensure high EE).

Listing 7.2: Algorithm 2 – a random change

```
double rnd = ((rand()%(param*2+1)) - param)/100;
// rnd is between -param/100 and +param/100
fatigue = fatigue + rnd;
```

3. The purpose of this algorithm is to ensure that when the WLU "stabilizes", so will the *fatigue* of a node, with the idea that a (nearly) optimal balance

[1]The reason fatigue to be incremented by param*2 is because WLU very rarely increases, due to the nature of the software implementation, as will be seen later in this chapter.

between EE and latency is found in the network. This algorithm is a learning function and is shown in Listing 7.3. The value of `param` can be seen here as a learning rate, set at the beginning of the simulation. The node changes its `fatigue` according to the change in `WLU`. This algorithm is similar to the first one, but its action is opposite and the magnitude of change in WLU now matters – the bigger the change in WLU, the bigger the influence on `fatigue`. In other words, the higher the energy efficiency of a node, the more active it should be in communication, so that the messages are delivered faster to the sink, i.e. latency is reduced.

Listing 7.3: Algorithm 3 – a learning function

```
fatigue = fatigue + (old_WLU − WLU)∗param);
```

Before we investigate the effects of the above algorithms on EE of the network, we have to explain how a node *acquires* the information needed for computing the WLU in the first place. This will be described in the next section.

7.3 Obtaining the WLU

Chapter 6 defined the wonderful life private utility of agent i for the joint action ζ of all agents as follows:

$$WLU_i(\zeta) = G'_i(\zeta) - G(CL_{S_i^{eff}}(\zeta))$$

where $G'_i(\zeta)$ is the estimated world utility for agent i and $G(CL_{S_i^{eff}}(\zeta))$ is the contribution of the agents that are not part of the guessed effect set S_i^{eff} of agent i. In fact, the *key* aspect in computing the node's WLU for the current software implementation is the computation of the node's effect set.

7.3.1 Obtaining the Effect Set

Our implementation of the size of the effect set is slightly different than the one shown in Figure 6.1, because the latter does not take into consideration the message flow of the network (there is no sink in the picture!). In our simulation, the message flow is always toward the sink, therefore the area that a node affects is not symmetrical in all directions, but elongated and shift toward the sink. Figure 7.2 illustrates a more realistic effect set and the one used in our simulation. By broadcasting, the nodes

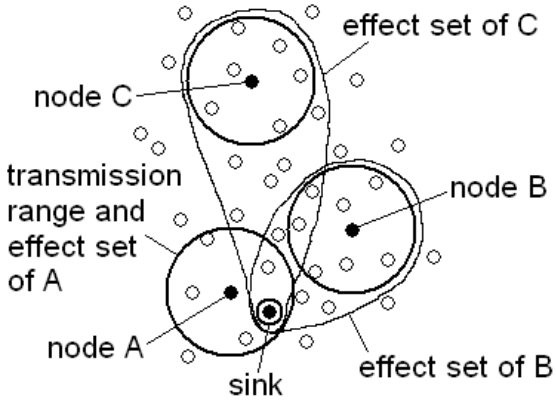

Figure 7.2: Effect set of nodes at different distances from the sink

exchange information that can allow them to compute their WLUs. In addition to other required data, every control packet contains information about the sender and its neighbourhood, so that when a node broadcasts a message, all neighbouring nodes can hear the following:

- The EE of the sender, so that neighbours can update the EE of *their* neighbourhood;
- The EE of the *original* sender (the one who generated the original message), so that nodes can compute the second term in the WLU (explained later in this chapter);
- The size and EE of the effect set of the sender, so that neighbours can update the size and EE of *their* effect set;

- The hop distance of the sender to the sink, so that neighbours can determine whether they should forward messages to the sending node or not;
- Remaining energy of the sender, so that neighbours can avoid this node in case its energy is low;
- Recent message "workload", indicating the amount of messages the node has forwarded recently, so that neighbours can leave it to "rest", if it has processed more than the other nodes in range.

Using the above information from neighbours, every node is able to compute in a straightforward way the EE of its *neighbourhood*, which is the average EE of all nodes within its transmission range. Once a node knows the EE of its neighbourhood, it can compute the size and EE of its guessed effect set S_i^{eff} in the following way:

1. If the sink is within transmission range of the node, that node's effect set size and EE is equal to the size and EE of its neighbourhood alone (see node A in Figure 7.2).
2. If the sink is not within range, the node adds the neighbour's effect set size with a portion of its neighbourhood size to update its own effect set size. (The "portion" will be explained below.) The EE of its effect set is obtained in a similar way – the node takes the normalized sum of the EE of the neighbour's effect set and a portion of its neighbourhood EE (see nodes B and C in Figure 7.2).

Listing 7.4 shows the rules each node uses to update its effect set size and EE. The algorithm for updating the effect set size and EE, shown in Listing 7.4, uses a variable, called `portion`, a value in the interval $[0, 1]$. It is defined as the ratio of the non-overlapping area in a circle, when two circles with equal radii overlap, to the area of one of them. A value of 0.5, for example, signifies that half of the area of one circle is covered by the other. The purpose of this value is to ensure that some of the neighbouring nodes will not be counted twice. For example, Figure 7.3 illustrates

that three of the neighbouring nodes of A are also neighbours of node B. So if node B was to simply add its own neighbours to the effect set size that node A broadcasted, the three common nodes will be counted twice – once as neighbours of A and once of B. Therefore, `portion` makes sure this does not happen, by taking only a *part* of the neighbourhood size of B.

Listing 7.4: Rules to compute the effect set size and EE of each node

```
/*
the node obtains the effect set size and EE of the
   sender and stores them in sender.effect_set_size
   and sender.effect_set_EE, respectively.
*/
if (Sink_within_range)
{
        my.effect_set_size = my.heighbourhood_size;
        my.effect_set_EE = my.heighbourhood_EE;
}
else // if the sink is not within range
{
        my.effect_set_size = (my.effect_set_size +
           (sender.effect_set_size +
           my.heighbourhood_size*portion))/2;
        my.effect_set_EE = (sender.effect_set_EE +
           my.heighbourhood_EE*portion)/2;
}
```

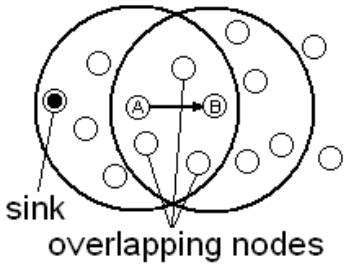

Figure 7.3: Overlapping nodes between the sender (A) and receiver (B)

The number of overlapping nodes depends on the transmission range, density of the network and the distance between the sender and receiver. Since nodes cannot determine this distance, nor the density of the network, they have to approximate the

number of overlapping nodes by taking only a fixed[2] fraction of their neighbourhood. The exact value of `portion` is set at the beginning of the simulation and equals to $\frac{3}{5}$, which approximates the non-overlapping area between two circles, with equal radii, at a radius distance from each other (as is the case in Fig. 7.3). In summary, this is how node B updates it's effect set size:

1. Node A sends a message to the sink and because of that broadcast, node B hears A's effect set size;
2. Node B assumes, that both nodes might have some common neighbours, but is unable to detect how many;
3. Node B updates its effect set size by adding only $\frac{3}{5}$ of its neighbourhood to the size, broadcasted by A.

For example, consider the network in Figure 7.3. Node A broadcasts the size of it's effect set to B. In this case, the effect set size of A is equal to the size of it's neighbourhood alone, because the sink is within range. So the effect set size of node A is 6. Node B has 7 neighbours, so it approximates that it's effect set size is:

$$\text{effect\_set\_size\_of\_B} = \text{effect\_set\_size\_of\_A} + \text{neighbours\_of\_B} \cdot \frac{3}{5} = 6 + 7 \cdot \frac{3}{5} = 10.2$$

as compared to the *real* effect set size of B, which is 10 (cf. Fig. 7.3). The effect set EE is computed in the same way. The node then takes an average between its old and new values for the size and EE of its effect set, to ensure correct approximations, because each neighbour can have a different effect set. However, to be able to compute the WLU, the node needs to have information about the world utility and the area, outside its effect set.

7.3.2 Obtaining the World Utility

According to our assumption in computing the effect set, each node affects only its neighbourhood and the nodes that lie on the message path to the sink. Of course, this

[2] assuming the nodes are not moving

is not true in the long run, where each node eventually affects all others, but since the influence is decreasing with distance, that influence is negligible for nodes outside the message path of a node. For this reason, we can limit the notion of "world" to only those nodes, that directly affect a node, or are directly affected by it. In other words, a node's world is the set of nodes, whose messages pass through that node, or who forward it's messages to the sink. In the extreme case – when a node is at the border of the WSN, its world will be equal to its effect set. This is because there are no nodes which can affect it, while the nodes that it affects constitute its effect set. For example, in Figure 7.4, the world for node A is the set of nodes W1, while for node C, the world is W2. Since nodes B and D are influenced by all nodes in W1 *and* W2, their world size and EE is that of the union W1 ∪ W2 of nodes. It is clear that the size of the worlds might be different, like in the one of node E, which consists of only 3 nodes.

In summary, every node propagates its *effect set* size and EE backwards (toward

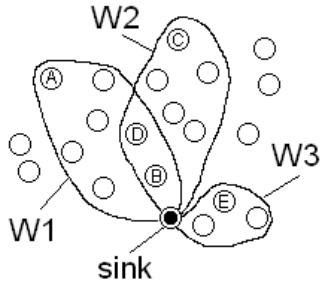

Figure 7.4: "World" of a node – the world of A is W1, while of B it is W1 ∪ W2

the end nodes of the network) so that the size and EE of the *world* can be approximated by the last hop nodes and then propagated forwards (in the direction of the sink) to provide the first term in the WLU – the estimated world utility $G'_i(\zeta)$. The second term in the WLU, $G(CL_{S_i^{eff}}(\zeta)$, is that of the area outside i's effect set. This is approximated by taking the average EE of all nodes, whose messages agent i is forwarding to the sink. Recall that the EE of the original sender is contained in every packet. This information is used by every node that forwards another node's

message in order to compute the average EE of the area outside it's effect set. This method can be applied, because the original sender is always outside the effect set of a forwarding node, other than its neighbours, due to the routing protocol used. For this reason, forwarding nodes can take the average EE of all original senders to approximate $G(CL_{S_i^{eff}}(\zeta)$, i.e. the contribution of the agents that are not part of the guessed effect set of agent i. Figure 7.5 clarifies the direction of information flow, used to compute the WLU. Note that the "backward flow" does not happen literally,

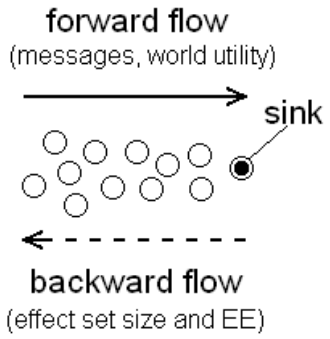

Figure 7.5: Forward and Backward flow of information, required to compute the WLU

i.e. nodes do not send information in this direction, but rather by the broadcasts of nodes (hence the dashed line). When a node sends a message, its higher hop neighbours hear the effect set size and EE of that node and later when *they* get to send a message, *their* higher hop neighbours receive that information, and so on, until it reaches the end of the network. Only then this information "transforms" to world utility and starts to flow forward.

It is interesting to notice that the value of the WLU for each node in fact resembles the effect set EE of that node, because of the way WLU is computed. If one takes all nodes that an agent affects or is affected by, which represents our world utility $G'_i(\zeta)$ and subtracts from it all nodes that affect this agent, which is the term $G(CL_{S_i^{eff}}(\zeta)$, one will be left with all nodes that are affected by this agent. This is in fact the effect set S_i^{eff} of that agent. Therefore, we expect to obtain similar results

when using the effect set EE as the private utility, compared to the case when the private utility is the WLU.

7.4 Summary

In this chapter we explained the practical implementation of the COIN solution method for WSNs, introduced in Chapter 6. We proposed three algorithms for private utility optimization, which are intended to increase the world utility by making nodes adopt mixed strategies when choosing their schedules. These three algorithms will be further discussed and tested in the next chapter. We also showed how the communication protocol can be used, so that nodes can obtain the components of the wonderful life private utility and update their knowledge about the performance of the other nodes in the system. Finally, we proposed that the WLU of a node should (closely) approximate the effect set energy efficiency of that node. In the next chapter, we will test this proposition experimentally.

Chapter 8

Experiments and Results

In this chapter, we will examine the performance of the three algorithms, introduced in Section 7.2 and will compare it to the scenario where nodes do not use any *learning algorithm.*

Chapter contents: Experiments, Results, Discussion.

8.1 Introduction

In this chapter we will show experimental results of the following scenarios:

0. Nodes use **no algorithm** to optimize their private utility. They are simply following their periodic schedules with equal number of timeslots for sleeping and listening. In this case, the value of the WLU is not used at all. This scenario serves as the reference point between the three algorithms.

1. Nodes use a **decision rule** (algorithm 1) to determine the amount of timeslots to sleep and to listen in their periodic schedules. A decreasing WLU (irrelevant of the magnitude) results in more listening, while increasing WLU makes the node sleep more. The purpose of using this rule is to check whether low energy efficiency is caused by control packet overhead.

2. Nodes use a **random change** (algorithm 2) in periodic schedules to try to achieve higher energy efficiency. Similar to Scenario 0, the nodes ignore the value of their WLU. Instead, they choose the duration of their regular sleeping and listening modes according to a random variable. This case compares the WSN scenario to the El-Farol Bar Problem and suggests that by using mixed strategies, agents can become more energy efficient.

3. Nodes use a **learning function** (algorithm 3) to find a periodic schedule that will ensure high values of energy efficiency. The more the WLU is decreasing, the more the node will tend to sleep, while increasing WLU results in more listening. The purpose of this case is to check whether low energy efficiency is caused mainly by idle listening and overhearing, rather than control packet overhead, as opposed to scenario 1.

In Section 8.2, we will continue with the description of the experimental setup and in Section 8.3 we will present the types of experiments performed, followed by the actual experiments and results in Section 8.4. Finally, we will discuss our findings in Section 8.5.

8.2 Experimental Setup

The WSN simulation was was specifically tailored for the purpose of this research. It was programmed in the CompC++ language, a component-oriented extension to C++ [40] and run in the COST simulation environment, a general purpose discrete event simulator [41].

The wireless networks, used in the experiments, were composed of nodes, aligned in a perfect **grid structure**, rather than being randomly distributed in the environment. Although this is rarely the case in real-life scenarios, the sensor network in our

simulation forms a grid with nodes at equal distances from each other. This is done for reasons of simplicity and to ensure that no nodes fall far from others and thus become disconnected from the network, when distributed randomly. Moreover, we use this structure to test the practical application of the COIN framework, rather than to propose an "out-of-the-box" practical solution. We believe that a grid structure will be suitable for the purposes of this research.

We specify that the distance between each node in the grid is 1 unit. The **antenna range** is equal for all nodes and is set to 1.5 units in radius, so that all neighbouring nodes in the grid are also neighbours in the WSN (see Figure 8.1). In this way, a node can have at least 3 neighbours (e.g. node A in the Figure), when at the corner of a network, and at most 8, when not at the border (e.g. node B in the Figure). The value of `fatigue` is initially set to 0.5, which makes the duration of listening and

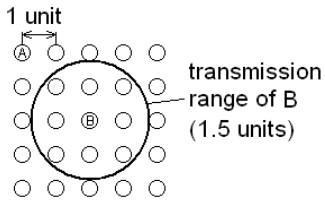

Figure 8.1: The grid structure of the WSN and the antenna range of a node

sleeping in the periodic schedules equal. We set the frame in a schedule to consist of 10 slots, which means that the periodic listening occurs in the first 5 slots and the sleeping – in the last 5 slots of the frame. Each node can store up to 50 messages in its queue, before starting to drop incoming messages. Upon failing to send a message (no receiver listening), a node will retry once, before giving up and going to sleep. Nodes can forward messages only to lower hop neighbours, i.e. to nodes that are closer to the sink, than they are. If more than one such nodes exist, the sender will choose the one that has processed fewer messages in the past frames. The information about the amount of processed messages is shared between nodes, so that each node keeps track of the number of processed messages of its neighbours.

8.3 Experiments

Each of the four scenarios in Section 8.1 were tested on two networks – one small (size 5×5 units with 25 nodes) and one big (size 10×10 units with 100 nodes). For comparison, the GlacsWeb project [42] used a sensor network of 9 nodes to monitor the behaviour of Briksdalsbreen glacier in Norway; while 32 nodes were used to monitor the habitat of birds on the Great Duck Island near Maine, USA [31]. We therefore consider that 100 nodes constitute a relatively large WSN.

In Section 7.3.2 we hypothesized that we will obtain similar results when we set the private utility of an agent to be its effect set EE, rather than its WLU. We will investigate this proposition, by running two more experiments on the big network, when agents try to optimize their private utility by using the same algorithms (1) and (3), but this time with the effect set EE as the node's private utility. In summary, the following experiments were performed:

1. No algorithm for private utility optimization, big network[1]. (reference point for experiments on the big network)
2. No algorithm for private utility optimization, small network. (reference point for experiments on the small network)
3. Algorithm 1, optimizing the WLU, big network.
4. Algorithm 2, big network.
5. Algorithm 3, optimizing the WLU, big network.
6. Algorithm 1, optimizing the WLU, small network.
7. Algorithm 2, small network.

[1]Here we switch the logical order of presenting the experiments, i.e. we present first experiments on a big network and then – on a small, so that our descriptions and discussions will later become more clear to the reader.

8. Algorithm 3, optimizing the WLU, small network.

9. Algorithm 1, optimizing the effect set EE, big network.

10. Algorithm 3, optimizing the effect set EE, big network.

These experiments are summarized in Table 8.1.

In each of the above experiments, we evaluated four criteria, which serve as the

	Big network		Small network	
	WLU	Eff. set EE	WLU	Eff. set EE
No Algorithm	#1		#2	
Algorithm 1	#3	#9	#6	-
Algorithm 2	#4		#7	
Algorithm 3	#5	#10	#8	-

Table 8.1: Types of experiments

reference point between the experiments. The four criteria are as follows:

1. The mean accuracy with which each node has determined the EE of the world, throughout the simulation. This value, "**known EE accuracy**", informs us how good the nodes have estimated the world utility. This value is important, because if that estimation differs from the true one significantly, we expect that the private utility will decrease. In other words, the estimated world utility $G'_i(\zeta)$ will differ significantly from the true world utility $G(\zeta)$. "Known EE accuracy" of 100% means that each node knows the exact value of the world EE.

2. The true EE of the world, "**true EE**". This shows the mean energy efficiency of nodes throughout the simulation, as measured by an external observer, rather than by nodes themselves. This is the value that nodes will try to improve, by changing their periodic schedules. Increasing values of "true EE" will result in

longer autonomous lifetime of the system, which is our main objective. "True EE" of 100% means that no energy is wasted by any node in the network.

3. The latency of the network. We have expressed the latency in terms of "**received messages**", which signifies the number of messages that the sink has received throughout the simulation runtime as a part of all generated messages. Here the intuition of latency is reversed, because higher number of "received messages" means low latency. While our primary objective is to increase the energy efficiency of the system, our secondary objective is to keep the latency low (or high value of "received messages"). When this value is 100%, it means that all generated messages are received by the sink within the duration of the simulation.

4. The battery life at the end of simulation. This value, "**battery life**", shows the mean amount of remaining battery for each node when the simulation ends. A value of 100% means that no node used any energy during the simulation. Since the starting battery life and energy dissipation are hardware dependent, we will use only relative values, i.e. we will base the performance of this value on the case where no algorithm is used for private value optimization.

All values are represented in percent and for all of them high percent means good system performance. We will base the obtained values from the experiments on the case where nodes use no algorithm to optimize their EE. As we explained in Section 7.2, each of the three algorithms are fine-tuned by a value, called `param`, which is set at the beginning of the simulation. This value is chosen independently and serves to optimize the performance of each algorithm for the specific scenario. In our experiments, we will show the performance of each algorithm for 20 different values of `param`. We believe that this amount of values will provide enough information about the performance of the algorithms for the two different settings (the small and the big networks) and will help the observer choose an admissible tradeoff between energy efficiency and latency of the network.

8.4 Results

8.4.1 Experiments #1 and #2

In the first two experiments, we measured the performance of the four criteria, specified above, when nodes use no algorithm to optimize their private utility. They are simply following their periodic schedules with equal number of timeslots for sleeping and listening in a frame. Experiment #1 was run on a big network (size 10×10 units with 100 nodes) and its results are shown in Figure 8.2. Experiment #2 was performed on a small network (size 5×5 units with 25 nodes) and the corresponding results are shown in Figure 8.3. As we discussed previously, the four most important criteria for our research are known EE accuracy (shown in blue), true EE (shown in red), received messages (shown in yellow) and battery life (shown in green).

The vertical axis of these graphs shows the performance of the four criteria in percent. For all of them high percent means good system performance. The horizontal axis shows the different values of `param` that were used in each experiment. In these two experiments, our graphs are constant, since nodes do not use any algorithm for private value optimization and therefore are independent of this value. If any of the graphs in the next experiments drops below this constant line, we consider that the corresponding criterion in the chosen algorithm performs worse for the specific value of `param` than if no algorithm was used. For example, if the true EE of the big network for algorithm 1 drops below the dashed red line for `param=1,2,3,4,5` in the first graph, we say that algorithm 1 in the big network performs worse than no algorithm for the first 5 values of `param`. In other words, algorithm 1 should not be initialized with any of those values, since it will have a negative effect on the network's true EE.

Section 7.3.2 explains how each node computes the known EE of the world. This criterion is significantly lower in Figure 8.3, because in the smaller network, most

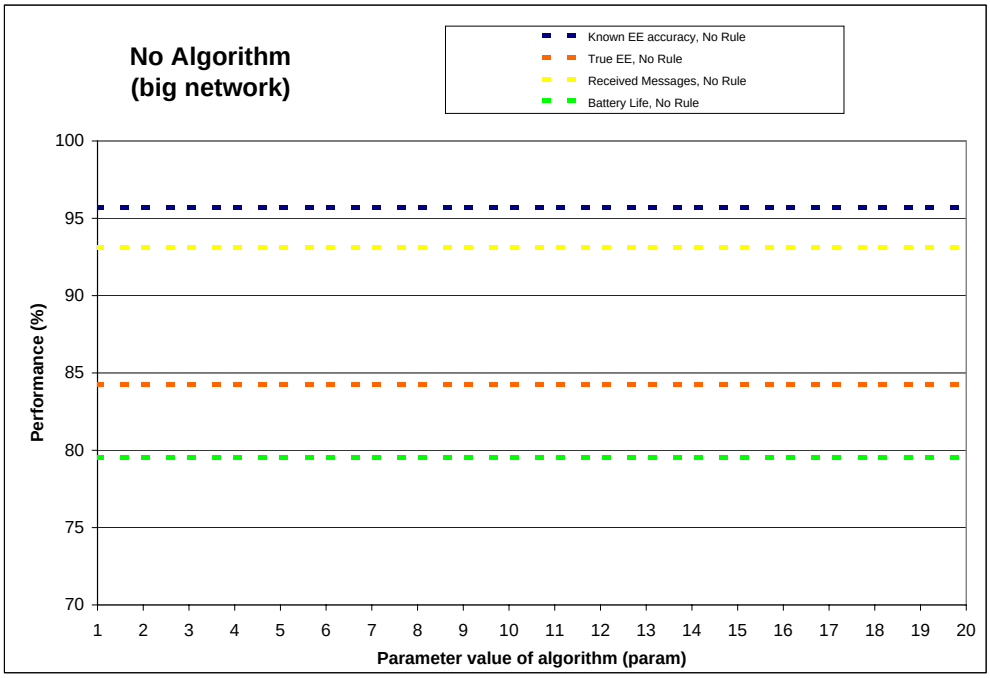

Figure 8.2: Results from Experiment #1, no algorithm for private value optimization, big network

of the nodes lie on the border of the WSN (64% of all nodes)[2] and therefore inner nodes will have many "different opinions" as to what the true EE of the network is. In other words, the effect set of the outer nodes is very small, which provides inaccurate estimation of the world utility to the nodes in that effect set. Therefore, the knowledge of the true EE for each inner node will differ signifficantly, leading to lower known EE accuracy for smaller networks. In the big network, this value is high, i.e. on average, nodes can estimate the true EE pretty accurately, because only 36% of all nodes[3] lie on the outer edge of the grid, which leads to a more precise world EE estimation.

[2] in a grid of 25 nodes, 16 lie on the border
[3] 36 out of 100 nodes

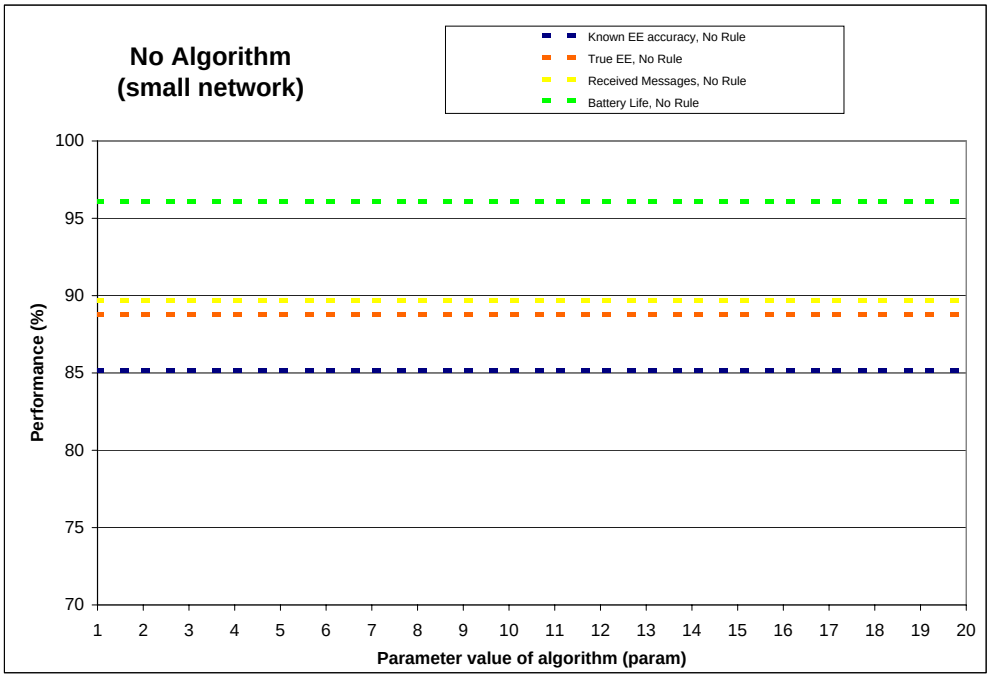

Figure 8.3: Results from Experiment #2, no algorithm for private value optimization, small network

The second criterion, the true EE of the world, is higher in the smaller network, because there are less nodes to generate messages and therefore, on average, a node forwards fewer packets than in the big network. In other words, a network with less nodes will process less messages and therefore will have higher EE, since energy loss happens mainly in the process of message forwarding. As the size of the network increases, we expect EE to drop due to this effect.

The ratio of received to generated messages in both experiments is comparable, because of the equivalent periodic schedules the nodes use in both networks. Ad-

ditionally, the duration of the simulation is relative to the network size, therefore, the bigger network was left to run much longer than the smaller, since much more messages had to be processed. We measured only 3% drop in received messages for the bigger network, which we consider an insignifficant difference.

The battery life is significantly higher in the second graph, as expected, because of the same reasons as before, i.e. smaller network will dissipate less energy on average, than a bigger one. The magnitude of difference, however, strongly depends on the simulation runtime, which is also shorter in the smaller network.

These results show that without an algorithm for private value optimization, a smaller WSN performs better than a bigger one, which confirms our initial prediction. We will now examine if this holds when nodes use an algorithm to optimize their own performance. Since energy dissipation is hardware dependent, our results from these two experiments will serve as a reference point between the three algorithms for private value optimization for the corresponding network size. In other words, the results from the next experiments should be regarded as *relative* to those in the first two experiments, rather *actual*, which one would obtain in a practical hardware network.

8.4.2 Experiments #3, #4 and #5

These three experiments are performed on the big network, each with a different algorithm for private value optimization. The performance of the four criteria in the corresponding algorithms is plotted as a solid line and is compared to the performance when no algorithm is used (plotted as a dashed line with the same colour). Figures 8.4 and 8.6 illustrate the performance of each criterion for the different values of `param` when nodes use the decision rule (algorithm 1) and the learning function (algorithm 3), respectively, to optimize their WLU. Similarly, Figure 8.5 shows the results of experiment #4, i.e. nodes use algorithm 2 in the big network. Each of these

algorithms is explained in Section 7.2.

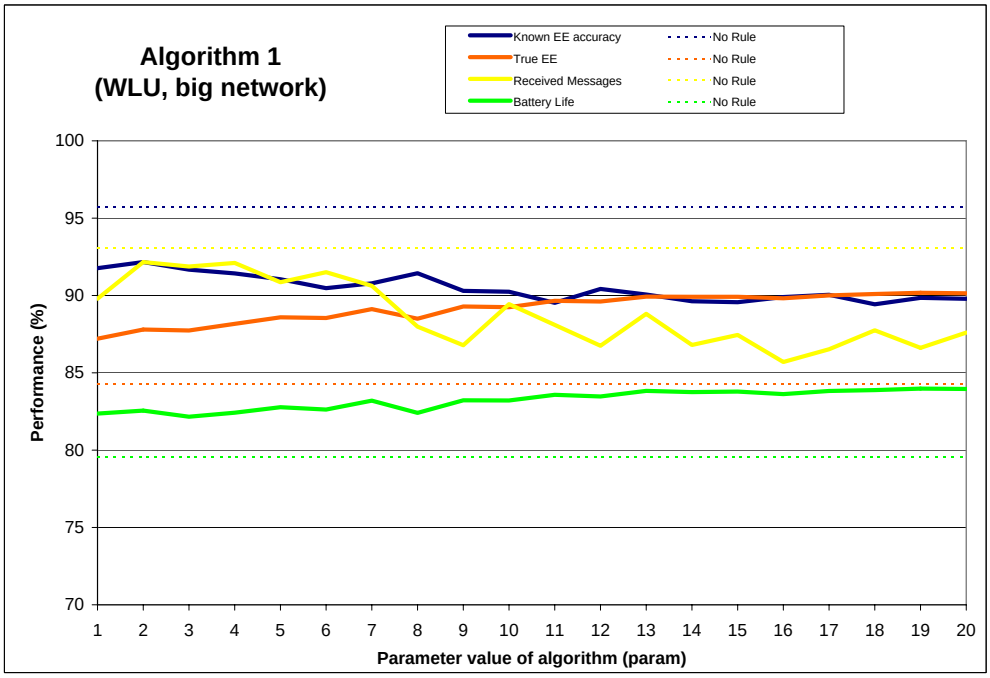

Figure 8.4: Results from Experiment #3, algorithm 1 optimizing the WLU, big network

Figure 8.4 shows that both the true EE and the battery life increase when nodes use algorithm 1 to optimize their wonderful life private utility, compared to the case when they use no algorithm. However, the amount of received messages at the end of simulation decreases, as well as the overall accuracy, with which nodes determine the EE of the network. The latter is not a problem, since what we are interested in is the *true* EE, which is higher than in the case where nodes use no optimization algorithm. The issue here is that latency is higher (i.e. lower amount of received messages for the duration of the simulation) than the one in experiment #1, because

algorithm 1 makes nodes sleep more, when their efficiency goes up, as is the case in Figure 8.4. We also notice that with increasing value of `param`, the number of received messages decreases, while the true EE increases, although these changes are relatively small. In other words, the EE and latency of the network change only slightly with different values of `param`, therefore we expect them to remain nearly the same when we extrapolate the graph for higher values of `param`. Since algorithm 1 was based on the suggestion that control packet overhead might be the bottle neck of communication in terms of energy efficiency, the results of experiment #3 show that this is not the main issue we should be concerned about, but we should not disregard it either. It even shows that when nodes are more active, the message forwarding is actually worse than when their listening and sleeping durations are equal. In summary, the energy waste by control packet overhead is reduced, at the cost of higher latency of the network.

In Figure 8.5 we see that although the true EE for `param=1` is the same as in experiment #1, the amount of received messages for the simulation runtime is a lot higher. In fact, nearly all generated messages are received by the sink within the duration of the simulation for the first 8 values of `param`, while the true EE and battery life are increasing significantly. This is because of the random sleeping schedules of nodes, or in other words, the mixed strategies they adopt. As we suggested in Section 7.2, this "random behaviour" of nodes prove to be evenly balanced – while some nodes are more active in communication (to ensure low latency), others sleep more (to ensure high EE). As the true EE and battery life increase with increasing values of `param`, received messages slightly decrease as a result of the bigger difference between the durations for listen and sleep, i.e. with high values of `param`, there is a higher probability that a node will listen a lot, thus overhear packets, or sleep a lot, thus slow down message forwarding. We believe, however, that a good tradeoff between EE and latency can be found using this algorithm, depending on the latency requirements of the observer.

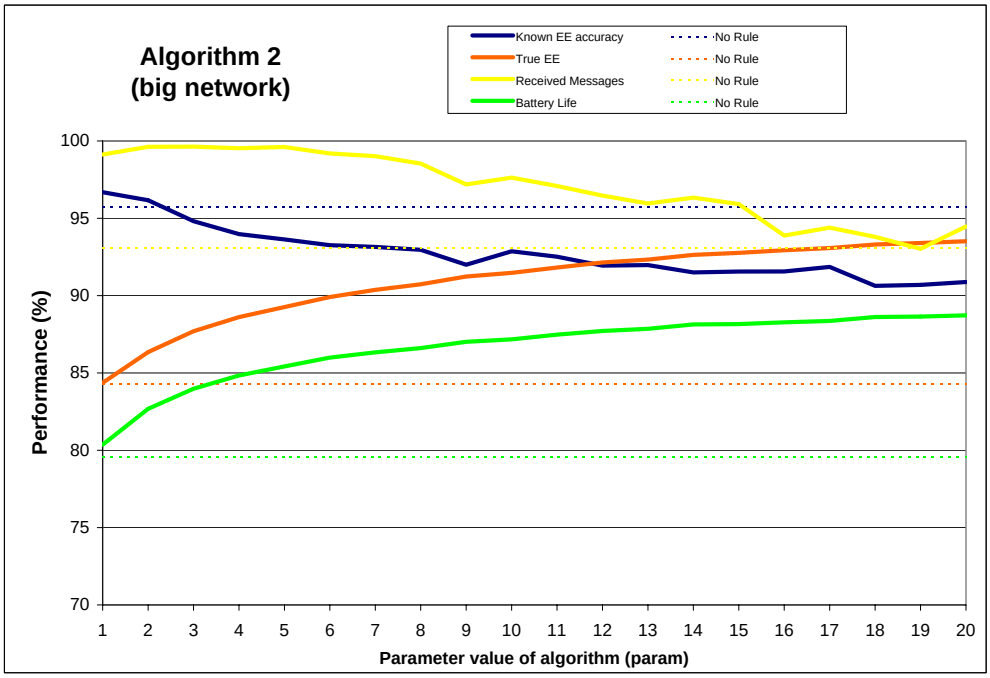

Figure 8.5: Results from Experiment #4, algorithm 2, big network

Experiment #5, shown in Figure 8.6, achieves the highest battery life and true EE of the network among the first 5 experiments, for `param > 10`. The received messages, however, are rapidly decreasing in this range. Thus, algorithm 3 makes nodes save much more energy, at the cost of very high latency. In Section 7.2 we explained that this algorithm is a learning function, where `param` is the learning rate. We can see from Figure 8.6 that a higher learning rate leads to higher system performance in terms of EE, because nodes are able to "stabilize" their WLU faster and thus find a (nearly) optimal periodic schedule that will ensure high EE of the network. As we expect, high EE leads to high latency as well, so one should again find an admissible tradeoff between these values. This algorithm makes energy efficient nodes more

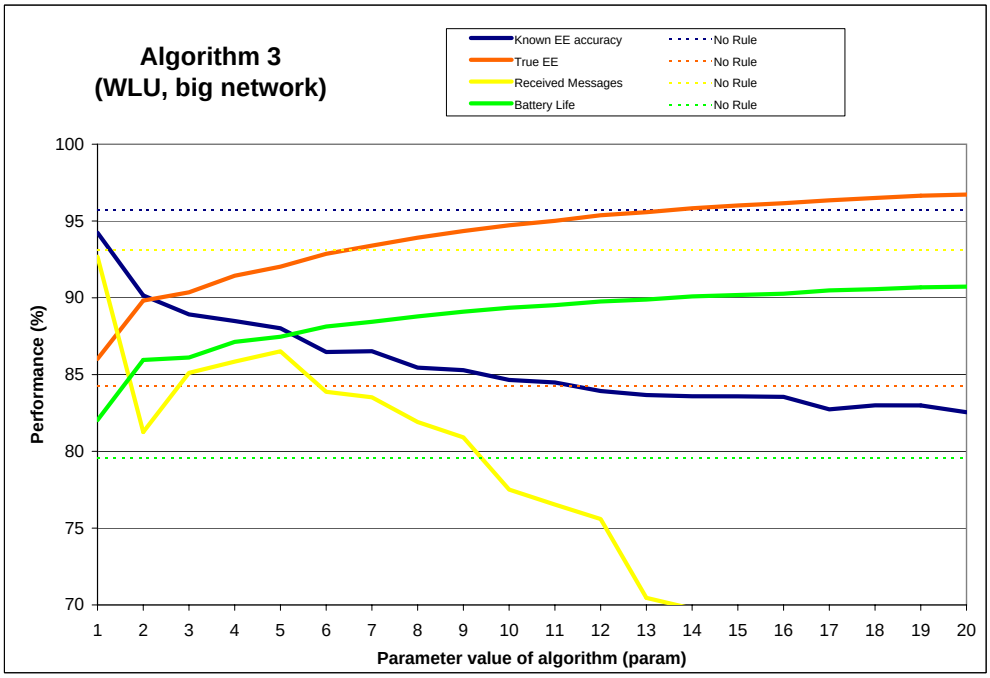

Figure 8.6: Results from Experiment #5, algorithm 3 optimizing the WLU, big network

active in communication, in order to deliver messages faster, but as we saw before, the more active the nodes are, the worse the latency is. However, if this latency is allowed by the observer, the WSN will have a very high autonomous lifetime when nodes use algorithm 3 with `param > 10` to optimize their WLU.

8.4.3 Experiments #6, #7 and #8

The next three experiments were used to test the behaviour of the three algorithms on a small network – 5×5 units with 25 nodes. Here again, each algorithm influences the performance of four criteria – known EE accuracy, true EE, received messages and battery life, which are plotted as solid lines, while the performances with no

algorithm are plotted as dashed lines with the same colour.

Experiment #6, shown in Figure 8.7, was performed on a small network with 25 nodes that used algorithm 1 to optimize their WLU. Since the network is small, the mean battery life at the end of simulation is significantly higher than the one in the big network. This effect is explained in Section 8.4.1. Here, our decision rule helped reduce the latency of the network, but was unable to significantly improve the true EE . In fact, all four criteria, except the received messages, remained relatively constant with changing `param` and with the same performance as in experiment #2. In other words, when all nodes use algorithm 1 to optimize their WLU, the latency of the network is lower, but the true EE remains the same as if they did not use any algorithm at all. In experiment #3 we saw that when the same algorithm is used in a big network, the true EE does improve, but the latency increases. So with increasing network size, algorithm 1 performs better in terms of energy savings, relative to the case where no algorithm is used, but increases the overall time for which a message travels to the sink.

In experiment #7 (Figure 8.8) we are able to see the same behaviour, although with a lower magnitude, for the four criteria, as in experiment #4, i.e. the true EE increases with increasing `param`, while almost all generated messages are received by the sink within the simulation runtime. In other words, when nodes adopt a random change pattern for their periodic schedules, the autonomous lifetime of the network is significantly increased *and* the duration for messages to reach the sink is greatly reduced. This random change pattern represents in fact the mixed strategies that agents adopt to solve our multi-agent problem, similar to the El Farol Bar problem (cf. Section 7.1). Since this algorithm does not depend on the performance of the individual agent, even with increasing network size, the true EE and latency of the system remain relatively unchanged, due to the even balance between sleeping and listening.

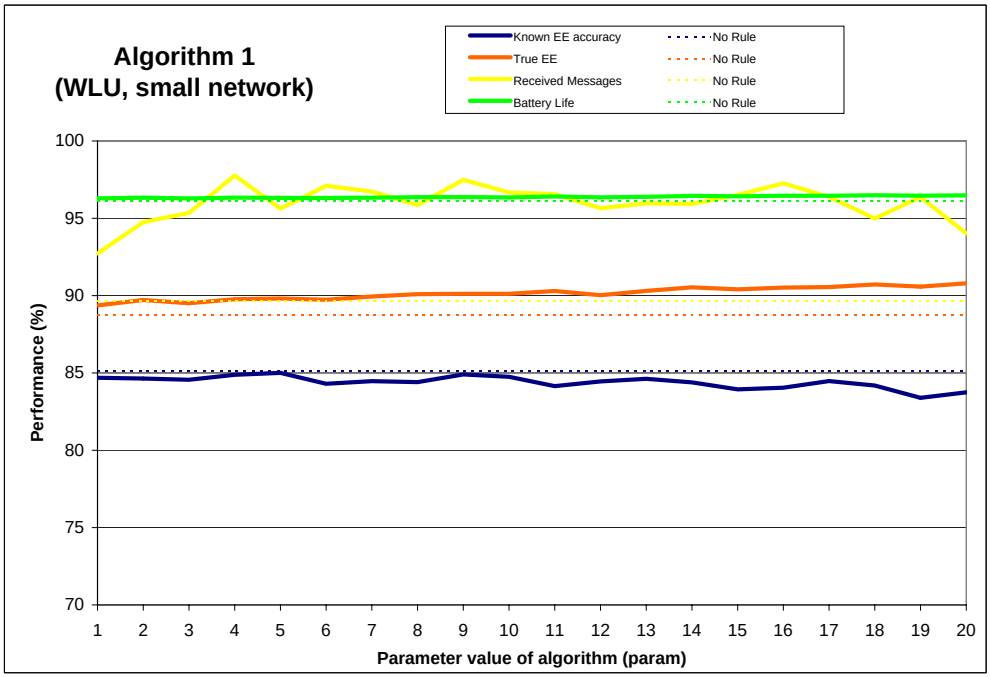

Figure 8.7: Results from Experiment #6, algorithm 1 optimizing the WLU, small network

The behaviour of the four criteria in experiment #8 (Figure 8.9) is similar to the one in experiment #5 – the true EE is increasing, while the number of received messages is decreasing rapidly for increasing values of `param`. When nodes adopt the learning function (algorithm 3), they are able to achieve the highest energy savings, compared to the other algorithms for the corresponding network sizes. The highest EE, however, comes at the price of very high latency. However, if this latency is tolerable, the WSN is able to achieve the highest autonomous lifetime than with any of the other algorithms. Moreover, the increasing network size does not change the performance of the system, i.e. the four criteria remain relatively unchanged. This is due to the

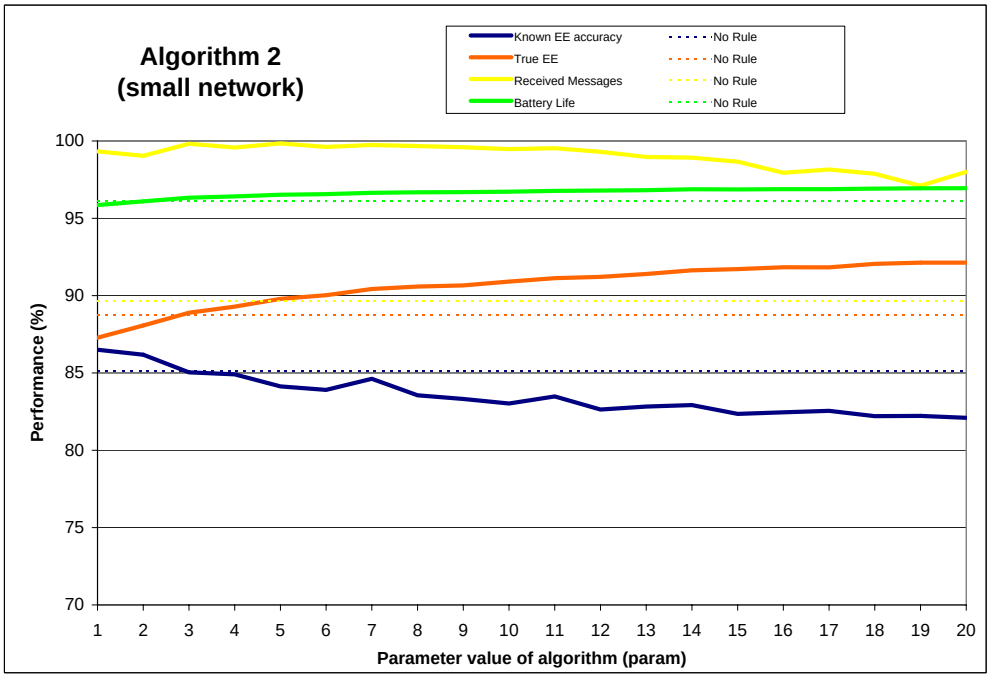

Figure 8.8: Results from Experiment #7, algorithm 2, small network

adaptive behaviour of the learning function with a high learning rate (i.e. high `param`) when more nodes are added to the network.

8.4.4 Experiments #9 and #10

These two experiments were used to test whether the performance of the agents, optimizing their WLU, will be the same as when they are optimizing their effect set EE instead. In Section 7.3.2 we argued that their performance should remain relatively unchanged, since the WLU represents in fact the effect set EE. Thus, the two private utilities can be seen as equal and therefore, experiment #9 should provide similar results to experiment #3, because they both use algorithm 1 to optimize their pri-

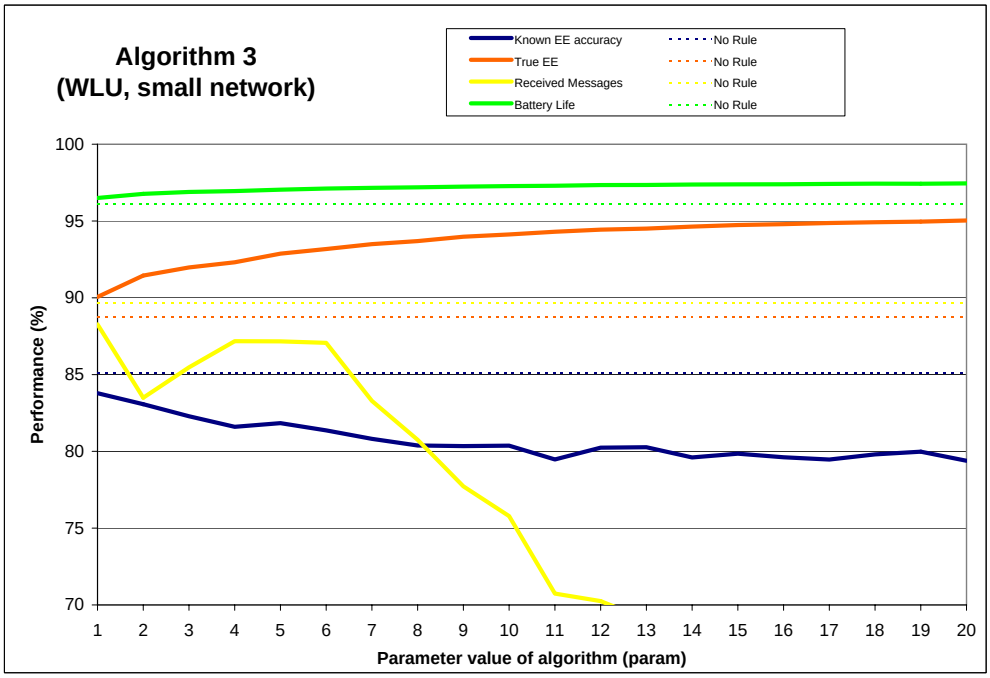

Figure 8.9: Results from Experiment #8, algorithm 3 optimizing the WLU, small network

vate value in a big network. The same goes for experiments #10 and #5, which use algorithm 3. It is clear that algorithm 2 cannot be tested here with the node's effect set EE as a private value, since the agent uses a random value to change its periodic schedules and therefore ignores it's private utility.

Experiment #9, shown in Figure 8.10 achieves higher true EE than experiment #5. In fact, the results of both experiments are far from comparable, contrary to what was initially expected. The extra high true EE results in extra high latency for increasing values of `param`. It can also be seen that the mean battery life at the end of simulation is a lot higher than the one in experiment #3. Since the only difference

between these experiments is the private value, we should seek the reason for the different results there (explained in Section 8.5).

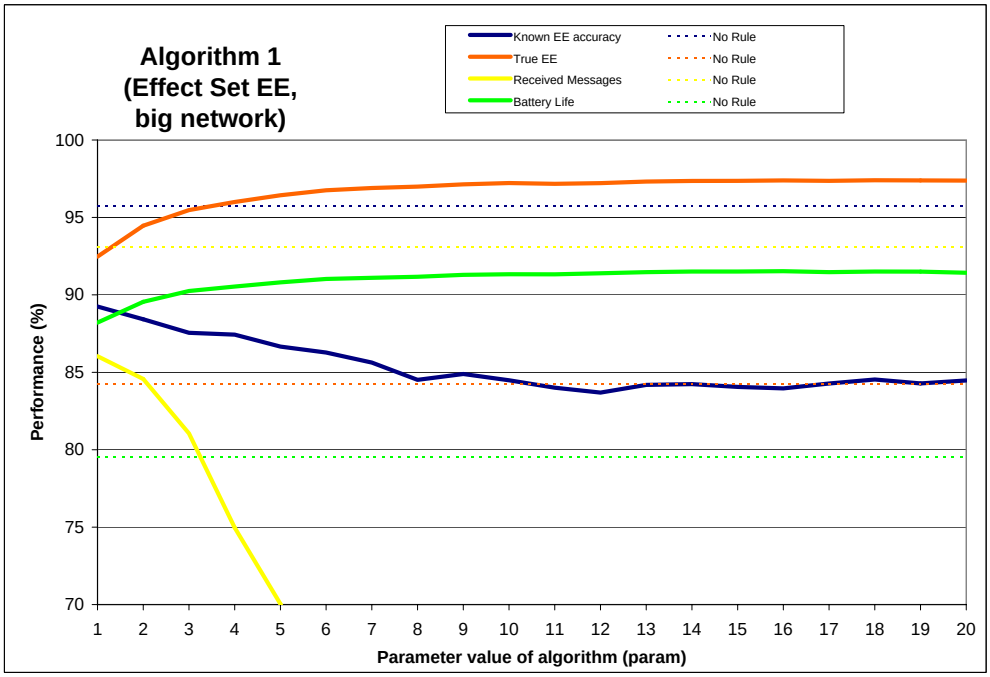

Figure 8.10: Results from Experiment #9, algorithm 1 optimizing the effect set EE, big network

Contrary to the above situation, experiment #10 showed results, comparable to those in experiment #5. The curves for the true EE, known EE accuracy and battery life in Figure 8.11 are similar to those in Figure 8.6, although a lot more noisy for the different values of `param`. One difference in the results of both experiments is that the latency of the network, with effect set EE as private value, is significantly lower for `param > 6` than when nodes use the WLU as their private value. The reason for this difference is explained in Section 8.5.

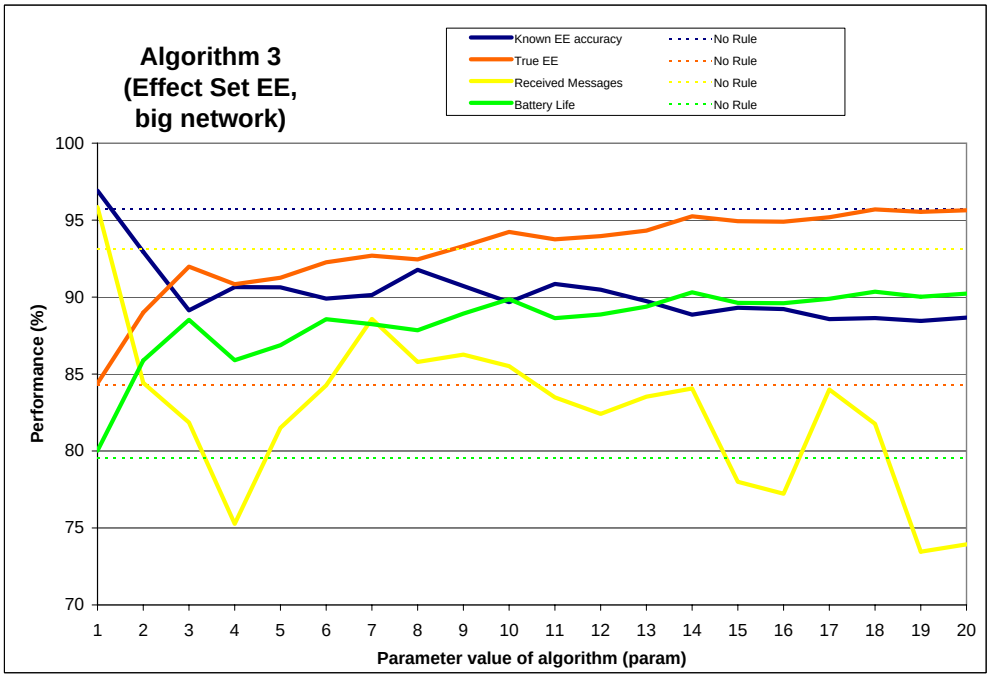

Figure 8.11: Results from Experiment #10, algorithm 3 optimizing the effect set EE, big network

8.5 Discussion

The above experiments were used to test the influence of three algorithms on the EE and latency of a small and big WSN, when nodes use two different private utility functions. Since our main objective is to increase the EE of the WSN, most of the experiments sacrifice latency in order to make nodes save more energy, rather than to try forward the messages faster.

The first algorithm – a decision rule, makes nodes more active, so that the number of retried transmissions is lower and thus the energy loss by control packet overhead is reduced. Different values of the parameter `param`, increase or decrease the rate

of change in the periodic schedules. When a large number of nodes are active, the message forwarding becomes worse, because many nodes try to send messages at the same time and thus the latency is increased. The energy savings by control packet overhead is most pronounced when nodes use the effect set EE as their private utility. When the WLU is used instead, the effect of the algorithm is weaker and therefore suitable for general purpose applications, because it achieves a good trade-off between EE and latency, without sacrificing any of them in the name of the other.

The second algorithm – a random change in the periodic schedules of nodes, makes nodes adopt mixed strategies by letting each node listen more or sleep more with a certain probability, thus achieving high EE and low latency of the network. The parameter value of this algorithm, `param`, defines the range, in which the duration of the listening and sleeping is chosen. The overall balance between sleeping and listening accounts for very fast message delivery and lower energy loss. This algorithm prioritizes the importance of latency and makes energy savings a secondary objective, therefore it is suitable for applications, where latency is crucial. This algorithm does not use the private utility of the agents and it is suitable for both small and big networks, as the results from both experiments #4 and #7 show.

The third algorithm – a learning function, tries to reduce idle listening and overhearing by putting nodes to sleep, if their efficiency is decreasing. Therefore, it sacrifices latency in order to improve the EE of the system. The parameter value of the algorithm, `param`, is the learning rate of this function. Higher values make nodes more energy efficient and slower in forwarding messages. The effects of the two different private utilities on this algorithm are comparable, although when nodes optimize their effect set EE, more messages are delivered to the sink within the simulation runtime. Experimental results show that the size of the network does not influence significantly the performance of this algorithm.

Most experiments show that our three algorithms are able to increase the EE of the network at the cost of increased latency. Different values of `param` make this tradeoff more or less pronounced, in order to make our approach applicable to a wide range of applications. For example, in habitat monitoring, low latency is usually admissible, since observations are collected over a long period of time and therefore, latency can be sacrificed in order to increase the autonomous lifetime of the system. Thus, observations can continue longer and the habitat of the observed species is not disturbed by the replacement of the depleted sensors. If such an application requires a small network, algorithm 3, with WLU as private utility, will be most appropriate, since it is able to achieve the highest energy savings for small networks by trading off latency. If the observed habitat is larger, it is better to use algorithm 1, with the effect set EE as private utility, since it is able to save more energy when the number of nodes is larger. On the other hand, most military applications require the network to respond promptly and therefore latency is crucial. In this domain, algorithm 2 will be most appropriate, since it shows very low latency, while still being able to increase the energy efficiency of the WSN. Moreover, the size of the network does not influence its performance significantly, therefore it can be used with any number of nodes.

In Section 7.3.2 we proposed that we will obtain comparable results, when nodes use their effect set EE as private utility, instead of the WLU, because both utilities refer to the same set of nodes – those that a node affects with its behaviour. However, experimental results show that this is not the case (cf. experiments #3 with #9 and #4 with #10). The reason for the difference in the results is the way both utilities are obtained.

The effect set EE is obtained when nodes, close to the sink, announce the EE of their neighbourhood, so that other nodes can add this value to their own neighbourhood and further propagate this result in the direction, opposite to the sink (cf.

Section 7.3.1). In this way nodes can update the EE of their effect set relatively quickly. This is because the information about the effect set travels a short distance, i.e. from nodes, close to the sink – to each node. Thus, the mean distance that the information about the effect set EE has to travel for all nodes, is 0.5 the length of the network. The least distance is when a node is next to the sink (i.e. 0 times the length of the network) and the largest distance is at the end of the network (i.e. 1 times the length of the network).

On the other hand, the components of the WLU are obtained in a more compli-

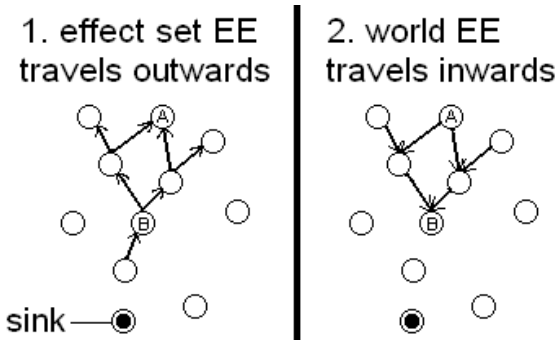

Figure 8.12: Obtaining the world utility

cated manner (cf. Section 7.3.2). The necessary information about the global utility component in the WLU has to travel from nodes, close to the sink, to the end of the network and back again to each node (see Figure 8.12). In other words, in Figure 8.12 node B updates its world EE component, when the effect set EE of nodes, close to the sink, is propagated outwards, to the furthest nodes in the network (e.g. node A), and from there on inwards, to that node. In this way, the mean distance that the information about the world EE component has to travel for all nodes, is 1.5 the length of the network. The least distance is when a node is at the end of the network (i.e. 1 times the length of the network) and the largest distance is next to the sink (i.e. 2 times the length of the network). Moreover, the world EE is only one of the two components in the WLU. The second component in the WLU of a node is computed by taking the mean EE of nodes, further away from that node in the direction, opposite to the sink. Thus, the computation of the WLU takes more

than double the time it takes for the effect set EE to be computed. Therefore, when nodes use the WLU, the estimation of the performance of their recent actions in the system, is delayed and therefore less accurate than when they use the effect set EE as their private utility. Less accurate estimations lead to lower EE, because nodes are unable to correctly evaluate the "goodness" of their actions in order to try to save more energy. For this reason, we believe that the effect set EE is more suitable to use as a private utility, since it is updated faster and therefore leads to more accurate estimation of the performance of the nodes. It is important to note here that with "more suitable" we compare *the way* each utility is computed in our simulation and not each utility *per se*, since theoretically, both utilities refer to the same set of nodes. The difference is in the *way of obtaining* each of the two values.

The different ways of obtaining the information for each private utility determines also the behaviour of the algorithms in nodes. The effect set EE is updated faster and nodes use less values to compute the mean, while the WLU takes more time and is evaluated over more nodes. Therefore, the effect set EE changes more frequently and with higher amplitude, than the WLU, which is more stable and changes only slightly across time. This is another reason for the difference in the shape of the curves in the figures of Section 8.4 for the two different utilities.

8.6 Summary

In this chapter we have run 10 experiments that evaluated the performance of two private utility functions and three algorithms for optimization of the behaviour of nodes in small and big networks. We identified the advantages and disadvantages of each algorithm and the purpose for which they can be used. Algorithms that improve the energy efficiency of nodes, while trading off latency, are suitable for environmental or habitat monitoring, where data is collected over a long period of time and thus

latency is a secondary concern. Algorithms that help forward messages faster can be used in domains, where latency is crucial, such as in disaster management or in military applications. We compared the effect of two different ways of obtaining the private utility of agents and concluded that the effect set EE provides more accurate evaluations of the behaviour of nodes which leads to better system performance.

Chapter 9

Conclusions

The aim of this research was to provide a general overview of Computational Mechanism Design and to study its practical application to the energy conservation problem in Wireless Sensor Networks.

Chapter contents: Conclusions, Challenges, Future Research.

9.1 Summary of Thesis

In the first part of this thesis we provided a theoretical overview of Game Theory (GT), Mechanism Design (MD) and Collective intelligence (COIN), investigated the relation between these fields and surveyed the state of the art in each of them. In Chapter 3 we showed why MD is referred to as Inverse Game Theory, by studying its relation to GT. In Chapter 4 we explained the relation of COIN to MD and the reasons why it is referred to as Learnable Mechanism Design.

In the second part of this research we surveyed the state of the art in Wireless Sensor Networks (WSNs), followed by the practical application of the COIN framework to this domain in Chapters 6 (analysis) and 7 (implementation). Finally we discussed experimental results from our simulations with different algorithms that aim to op-

timize the performance of individual nodes in order to increase the energy efficiency of the system. We evaluated the performance of these algorithms and discussed the trade-off one should make between energy efficiency and latency in the network.

9.2 Main Conclusions

From the first part of this research we can conclude that Mechanism Design is an approach to Game Theory that models the behaviour of agents in a perspecive, opposite to Agent Design. Agent Design takes the rules of the game as given and searches for a strategy, where deviation is not rational, so that the behaviour of rational agents can be predicted. Mechanism Design, on the other hand, models the private information of agents and designs the rules of the game, so that the desired system outcome can be reached, despite the self-interest of agents.

Similarly to Mechanism Design, Collective Intelligence searches for a suitable private utility function, so that when selfish agents learn to optimize it, they will accomplish the designer's goal. Because the essence of both approaches is the same and the fact that agents learn to optimize their performance, Collective Intelligence is often referred to as Learnable Mechanism Design.

Our main conclusion from the second part of this research is that nodes in WSNs are able develop an energy saving behaviour on their own, when using the Collective Intelligence framework, proposed by Wolpert et al. The extent, to which they can develop this behaviour depends mainly on the learning algorithm implemented and on the size of the network. Additionally, the accuracy, with which they approximate their information about the system further influences their performance – more accurate approximations lead to better performance.

Experimental results confirmed our hypothesis that the energy efficiency of the network is directly proportional to its latency, i.e. higher energy efficiency leads to higher latency. When nodes save more energy, for example by sleeping more, the time it takes for messages to reach the sink also increases. However, the magnitude of this influence once again depends on the learning algorithm, adopted by nodes.

9.3 Challenges and Future Research

Our research opens a huge amount of possibilities for future improvements, not only in the area of Artificial Intelligence (AI) alone, but also in numerous other fields that have influence on our approach. Here we present some of those possibilities for future research.

From an **AI** perspective:
During this research, we encountered a number of challenges, concerning the design and implementation of the COIN framework in the WSN simulation environment. One of the major challenges was the computation of the Wonderful Life Utility (WLU), because sensor nodes are restricted to obtaining only local information, while the WLU requires knowledge about the global system. We reduced the effect of this problem by including additional information in the control packets that nodes exchange during communication, which helps each node estimate its performance in the system. However, depending on the size and topology of the WSN, the accuracy of this estimation may be reduced, which leads to lower performance of nodes. Therefore, some of the challenges for future research are as follows:

- Optimize the way the WLU is computed, which will increase the overall performance of the learning algorithms, because nodes will be able to estimate the WLU with higher accuracy.

- Once the WLU is approximated in a better way, one could then compare the performance of nodes when they adopt the "new" WLU versus the case when they use effect set EE, as we have investigated in Chapter 8.

- Additionally, one could experimentally prove whether our approach of ignoring distant nodes in the computation of the WLU is correct, i.e. whether limiting the notion of "world" affects the WLU approximation in a negative way (cf. Section 7.3.2).

- Other areas for future research include testing additional learning algorithms and obtaining a better trade-off between energy efficiency and latency.

From a **network design** perspective:

- One could test what influence the existence of multiple sinks has on the performance of the WSN, since we expect that in future real-life applications, sensor nodes will be held by different stakeholders, serving different purposes. Each stakeholder might have different preferences about the position of his or her own sink. In what way will this affect the performance of the network?

- Crucial are experiments with different topologies of the network (e.g. narrow and long network, versus a more circular one, etc.), since the performance of the learning algorithms will change significantly. It is important to investigate the influence of the network's topology on the ability of nodes to save energy and forward messages faster to the sink(s).

- Although our WSN simulation tool already supports, among others, dynamic topology of nodes, we have not tested such a scenario. It would be a challenging task to work with a more realistic WSN simulation, by letting nodes move (e.g. simulation of a WSN in a water basin), fail (e.g. hardware malfunction or physical damage), or be dynamically added or removed from an already running system. Such scenarios are closer to real-life WSNs and therefore their results

will provide a more realistic picture of how our approach will perform in a real-life system.

From a **hardware design** perspective:

- An interesting possibility is to experiment how the performance of the system changes when nodes have more than one antenna, e.g. a short range and a long range one, or one for polling and one for transmission, etc.

- Additionally, one could experiment with more complex functionality of nodes, e.g. when they are able to detect signal strength and thus determine their distance to the sender. This could improve the routing of messages and therefore increase energy efficiency and decrease latency of the network.

- The choice of the hardware components is also crucial, since it will influence the performance of the learning algorithms (in a way yet to be experimented) and therefore might lead to higher energy efficiency of the whole system. It might also pose new constraints on the computational powers of nodes and therefore require alternative software design.

From an **internet technology** perspective:

- Improving the routing and/or the communication protocols will result in a more energy efficient network with lower latency. Optimizing these protocols could also lead to a better performance of the learning algorithms themselves, due to the better communication between nodes.

- Currently, synchronization of the network is a bottleneck for the energy efficiency of the system, since it uses huge amounts of energy and therefore the efficiency of each node drops significantly during the this phase. In our solution, the importance of synchronization is acknowledged, but currently ignored. Improving this area will lead to more efficient and robust solutions.

Bibliography

[1] Y. Narahari, *Game Theory*, lecture notes, Department of Computer Science and Automation, Indian Institute of Science, 2007 (unpublished).

[2] F. Vega-Redondo, *Economics and the Theory of Games*, 2001.

[3] M. Flood, *Some experimental games*, 1952, research memorandum RM-789.

[4] M. Dresher, *The Mathematics of Games of Strategy: Theory and Applications* (Prentice-Hall, 1961).

[5] T. C.-C. Tan and S. R. da Costa Werlang, "The Bayesian Foundations of Solution Concepts of Games," Journal of Economic Theory **45,** 370–391 (1988).

[6] J. Nash, "Equilibrium points in n-person games," In *Proceedings of the National Academy of Sciences*, (1950).

[7] D. Parkes, Ph.D. thesis, *Iterative Combinatorial Auctions: Achieving Economic and Computational Efficiency*, Department of Computer and Information Science, University of Pennsylvania, 2001.

[8] C. Schelling, *The Strategy of Conflict* (Harvard University Press, 1960).

[9] K. Prestwich, "Game Theory," Technical report, Department of Biology, College of the Holy Cross, Worcester, MA USA 01610 (1999) .

[10] J. M. Smith and G. R. Price, "The Logic of Animal Conflict," Nature **246,** 15–18 (1973).

[11] J. Weibull, "What Have We Learned From Evolutionary Game Theory So Far?," Technical report, Stockholm School of Economics and the Resarch Institute of Industrial Economics (2002) .

[12] J. M. Smith and G. A. Parker, "The Logic of Asymmetric Contests," Animal Behaviour **24,** 159–175 (1976).

[13] R. Selten, "Evolutionary stability in extensive-form two-person games - correction and further development," Mathematical Social Sciences **16,** 223–266 (1988).

[14] P. D. Taylor and L. B. Jonker, "Evolutionary Stable Strategies and Game Dynamics," Mathematical Biosciences **40,** 145–156 (1978).

[15] H. Gintis, *Game Theory Evolving* (Princeton University Press, 2000).

[16] K. Tuyls, P. Hoen, and B. Vanschoenwinkel, "An Evolutionary Dynamical Analysis of Multi-Agent Learning in Iterated Games," Autonomous Agents and Multi-Agent Systems **12,** 115–153 (2006).

[17] V. Conitzer, Ph.D. thesis, *Computational Aspects of Preference Aggregation*, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, 2006.

[18] D. Parkes, "On Learnable Mechanism Design," Technical report, Division of Engineering and Applied Sciences, Harvard University) .

[19] J. Lenting, "Drawbacks of Pareto Optimality, Strategy Proofness and Incentive Compatibility in the context of MAS mechanism design," Technical report, MICC-IKAT, University of Maastricht (1999) .

[20] W. Vickrey, "Counterspeculation, auctions, and competitive sealed tenders," Journal of Finance **16,** 8–37 (1961).

[21] R. Myerson, "Bayesian Equilibrium and Incentive Compatibility: An Introduction," Social Goals and Social Organization pp. 229–259 (1985).

[22] E. Muller and M. Satterthwaite, "Strategy-Proofness: The Existence of Dominant-Strategy Mechanisms," Social Goals and Social Organization pp. 131–171 (1985).

[23] A. Gibbard, "Manipulation of voting schemes: A general result," Econometrica **41,** 587–602 (1973).

[24] T. Groves, "Incentives in teams," Econometrica **41,** 617–631 (1973).

[25] E. Clarke, "Multipart pricing of public goods," Public Choice **11,** 17–33 (1971).

[26] D. Wolpert and K. Tumer, "Collective Intelligence for Control of Distributed Dynamical Systems," Europhysics Letters **49,** 6 (2000).

[27] D. Wolpert and K. Tumer, "An Introduction to Collective Intelligence," Technical report, NASA Ames Research Center (2008) .

[28] K. Tuyls, Ph.D. thesis, *Learning in Multi-Agent Systems, An Evolutionary Game Theoretic Approach*, Computational Modeling Lab, Vrije Universiteit Brussel, Belgium, 2004.

[29] N. Lemmens, Master's thesis, MICC-IKAT, University of Maastricht, 2006.

[30] K. Martinez, J. Hart, and R. Ong, "Environmental Sensor Networks," IEEE Computer **37,** 50–56 (2004).

[31] A. Mainwaring, J. Polastre, R. Szewczyk, D. Culler, and J. Anderson, "Wireless Sensor Networks for Habitat Monitoring," In *Proceedings of the 1st ACM International workshop on Wireless Sensor Networks and applications*, pp. 88–97 (2002).

[32] G. Simon, M. Maróti, and A. Lédeczi, "Wireless Sensor Networks for Habitat Monitoring," In *Proceedings of The Conference on Embedded Networked Sensor Systems*, pp. 1–12 (2004).

[33] F. Zhao and L. Guibas, "Wireless Sensor Networks," Elsevier (2004).

[34] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "A Survey on Sensor Networks," IEEE Communications Magazine (2002).

[35] P. Beyens, K. Tuyls, K. Steenhaut, and A. Nowé, "MacCOIN: Medium Acces Control for Wireless Sensor Networks based on Collective Intelligence," In *Proceedings of GTDT'04, held at the 3th International Conference on Autonomous Agents and Multi-Agent Systems*, pp. 1–12 (New York, USA, 2004).

[36] W. Ye, J. Heidemann, and D. Estrin, "An Energy-Efficient MAC Protocol for Wireless Sensor Networks," .

[37] J. Haartsen, "The Bluetooth radio system," IEEE Personal Communications Magazine pp. 28–36 (2000).

[38] D. Wolpert and K. Tumer, "Collective Intelligence, Data Routing and Braess' Paradox," Journal of Artificial Intelligence Research **16,** 359–387 (2002).

[39] A. Bell and W. Sethares, "The El Farol problem and the internet: Congestion and coordination failure," In *Proceedings of The Fifth International Conference of the Society for Computational Economics*, (Boston, MA, 1999).

[40] CompC++, "A Component-oriented extension to C++," http://www.cs.rpi.edu/c̃heng3/compc++/ (Accessed February, 2008).

[41] G. Chen and B. Szymanski, "COST: Component-oriented simulation toolkit," In *Proceedings of the 2002 Winter Simulation Conference*, (2002).

[42] K. Martinez, R. Ong, and J. Hart, "Glacsweb: a sensor network for hostile environments," In *The First IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks*, (2004).