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

# Coordinating Teams of Mobile Sensors for Monitoring Environmental Phenomena

by Ruben Stranders

Supervisors: Professor Nicholas R. Jennings and Doctor Alex Rogers
Examiners: Professor Vladimiro Sassone

A mini-thesis submitted for transfer from MPhil to PhD

June 8, 2009

UNIVERSITY OF SOUTHAMPTON

ABSTRACT

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

A mini-thesis submitted for transfer from MPhil to PhD

by Ruben Stranders

Mobile wireless sensors can play a vital role in achieving situational awareness in uncertain and changing environments by keeping track of environmental phenomena, such as temperature, gas concentration and radiation, that exhibit spatial and temporal correlations. Examples of such environments are commonly found in disaster response, where the safety and effectiveness of response units critically depends on the accuracy of estimation of the state of the world. In these environments, mobile sensors operating in a team can improve situational awareness by offering a high sensing resolution in a timely and efficient way. In order to do this efficiently, however, they need to coordinate their movements. This coordination is a challenging task, since the sensors operate in an environment that is highly uncertain and dynamic, have a limited perception of their surroundings, and have limited communication with adjacent sensors. Consequently, coordination mechanisms need to address the challenges involved in maximising the collective information gain of the entire team, in the presence of uncertainty and different world views. Previous work in this area has focused on the use mobile and fixed wireless sensors for environmental monitoring, but fails to provide a principled online, decentralised coordination mechanism for such settings.

In this report, we study the challenge of coordinating teams of mobile sensors for monitoring environmental phenomena. In order to do so, we review the literature on wireless (mobile) sensor networks, information processing, target tracking, and localisation and mapping. In particular, we focus on the key concept of adaptive sampling, which encompasses a set of techniques that aim to maximise information gain subject to movement constraints and the limited resources at a sensor's disposal. Based on this review, we present a general architecture for a sensor that makes a clear distinction between information processing, information valuing and maximising information gain.

In more detail, we show that the state of the art in adaptive sampling falls short of providing robust, scalable, decentralised coordination algorithms. To address these shortcomings, we develop two online, decentralised coordination algorithms for monitoring spatial phenomena. The first algorithm operates in an *un-negotiated coordination* mode, in which coordination is achieved exclusively through the exchange of observations; sensors need not coordinate (negotiate) about the actions they are about to take, but base their decisions solely on the picture of the state of the environment that they compiled using their own observations and those received from their neighbours. This algorithm is based on two techniques found in previous work. Firstly, Gaussian process regression (Rasmussen and Williams, 2006; Osborne et al., 2008), which is used for processing the raw observations obtained by the sensors and for predicting unobserved measurements. Secondly, myopic information-theoretic control (as found in Grocholsky (2002)), which is used for maximising the informativeness of the samples that are obtained by moving the sensors to locations where the environment is more uncertain.

The second algorithm extends the first by adding a negotiation stage, which results in *negotiated coordination*. This algorithm is based on the max-sum message passing algorithm for decentralised control (Farinelli et al., 2008), which allows the sensors to maximise a team objective function in a decentralised fashion. To make the max-sum algorithm suitable for solving the mobile sensor coordination problem, we develop two pruning algorithms that drastically reduce the amount of computation required. These pruning algorithms are generic in the context of applying the max-sum algorithm, and are thus not limited to the mobile sensor setting. Finally, we extend the negotiated algorithm for sensors that are characterised by continuous control parameters (for example their heading and velocity). To this end, we generalise the discrete max-sum algorithm to the continuous case in which the interactions between sensors are characterised by continuous piecewise linear functions.

# Contents

# List of Figures

# List of Algorithms

# Nomenclature

| | |
|---|---|
| $\mathcal{S}$ | A collection of sensors |
| $S_i$ | An individual sensor |
| $M$ | The number of sensors present in an environment |
| $\mathcal{G} = (V, E)$ | A graph encoding the physical layout of an environment, where $E$ encodes the possible movements between locations $V$ |
| $\mathcal{P}$ | A scalar field on one temporal and two spatial dimensions |
| $adj_{\mathcal{G}}(v)$ | The set of vertices adjacent to $v$ in graph $\mathcal{G}$ |
| $l_t^i$ | The location of sensor $S_i$ at time $t$ |
| $o_t^i$ | The measurement taken by sensor $S_i$ at time $t$ |
| $a_t^v$ | The actual measurement at location $v$ at time $t$ |
| $p_t^v$ | The predicted measurement at location $v$ at time $t$ |
| $\mathcal{X}_{v,t}$ | A continuous random variable associated with measurement at location $v$ at time $t$ |
| $U_i$ | The utility function of sensor $S_i$ |
| $p_i$ | The movement variable of sensor $S_i$ |
| $Q_{p_n \to U_m}(p_n)$ | A message from variable $p_n$ to function $U_m$ |
| $R_{U_m \to p_n}(p_n)$ | A message from function $U_m$ to variable $p_n$ |
| $\Delta^n$ | An n-dimensional simplex |

.

# Chapter 1

# Introduction

In disaster response, and many other applications besides, situational awareness (SA) is of critical importance. Understanding what is happening and how the situation is changing allows decision makers to choose the appropriate response to achieve their mission goals. In our research, we intend to establish or improve situational awareness by developing *coordination algorithms* for teams of mobile sensing robots or *mobile sensors* (see Figure 1.1 for examples) in order to provide a good coverage of their environment and to provide high sensing resolution of relevant events. In general, accurate and up-to-date SA is especially important in uncertain environments that are subject to rapid change and where the consequences of making decisions based on wrong or incomplete information can be disastrous.

In more detail, a commonly accepted definition of SA is "the *perception* of elements in the environment within a volume of time and space, *the comprehension* of their meaning, and the *projection* of their status in the near future" (Endsley, 1995). Focusing on the italicised words, we can distinguish between three stages of situational awareness: (i) perception, (ii) comprehension, and (iii) prediction. Let us illustrate these three levels in terms of a disaster response scenario: sensors achieve the first stage by collecting raw measurements of the temperature, gas concentration, and any other relevant environmental variables. This gives the sensors a low level of awareness of the current state of the environment, the relevant phenomena within it, and their dynamics. The second level of SA is achieved when the sensors attempt to comprehend their surroundings by detecting patterns in their observations, and determining the relevance of these observations in terms of the goals of the mission of the emergency response units. An example of a relevant observation is the detection of the presence of flammable gas, which could affect the mission goals significantly. The third and final stage of SA is achieved when the robots project their current picture of the world into the future. For instance, a robot that detects an increase in gas concentration is not only capable of determining that it is flammable (which is achieved in the comprehension stage), but also that there is an imminent risk of explosion.

(a) A mobile ground robot

(b) The Mars rover

(c) The Predator UAV (Unmanned Aerial Vehicle)

(d) The Hummingbird UAV

FIGURE 1.1: Examples of mobile sensors.

Given this example, we believe that the use of teams of wireless mobile sensors can contribute to great extent to the improvement of SA in dynamic environments,[1] since they are capable of delivering timely, detailed and relevant information to the decision makers. Furthermore, since they operate completely autonomously, they require little human intervention, and as such do not claim scarce human resources. Most importantly, though, the use of mobile sensors reduces the need of exposing humans to hostile environments (for example, Figure 1.1(b) and (c)), since they not provide vital information to decision makers, but can also perform dangerous exploration tasks on their behalf. As a result, the risk of injury and loss of life can be reduced.

Now, the reason for having these sensors operate in a team, instead of just as a collection of individuals, is twofold. First of all, by coordinating their movements to converge on the location of important events, the sensors are able to improve sensing resolution of those events, while preventing redundant coverage of less important events. Second of all, it allows the sensors to share information about their environment, which improves their understanding of it, and increases their ability to perform their sensing tasks. So,

[1] In fact, this vision is shared by both the ALADDIN project (`http://www.aladdinproject.org`) and the Centre for Robot Assisted Search & Rescue (`http://crasar.csee.usf.edu`), whose missions are to use autonomous robots for information gathering in disaster response scenarios.

as a result of the team coordination, the collective resources of the sensors are more efficiently exploited than in a non-cooperative, individual mode.

However, coordination within such a team of sensors creates a number of challenges. In particular, these challenges arise from the fact that each sensor is likely to have a different perspective of its environment and the actions of the other sensors, since the observation and communication range is limited in most practical scenarios. Specifically, this means that each sensor needs to be able determine which information to exchange with its neighbours, and to coordinate with them about when and where to move.

Against this background, the goal of our research is to develop algorithms for coordinating a team of multiple mobile sensors to improve situational awareness. More specifically, we focus on sensors that monitor environmental conditions in highly dynamic and uncertain scenarios, such as those found in disaster response. As such, the sensors will provide decision makers with the information they need to understand what is happening. More specifically, the informal discussion above gives rise to two functional requirements:

**Accuracy** The picture that is compiled by the sensors should provide accurate situational awareness. As discussed above, this implies that they should be able to detect the spatial and temporal patterns of the environmental phenomena in their environment.

**Relevance** The sensors should be able to detect the most relevant events in the environment. That is to say, the events that have significant influence on the success of the mission should be given more attention.

Apart from these functional requirements, we intend to develop algorithms that give a team of mobile sensors the following non-functional properties as well:

**Adaptiveness** The scenarios for which the mobile sensors are intended are characterised by their dynamism and inherent uncertainty. Consequently, the sensors will have limited knowledge of the prevalent conditions in the mission area before deployment. They must, therefore, be able to continuously adapt to the changing environment to provide up-to-date SA, and determine what to do next in order to improve SA. This involves continuously revising a model of the characteristics of their environment to best reflect the observations that have been made so far, and being able to evaluate the outcomes of possible decisions.

**Robustness** Clearly, in the hostile scenarios associated with disaster response, physical robustness of the sensors is essential. For instance, they should be able to withstand high temperatures and impact from debris. However, and more relevant in the context of our research, the sensors should also be able to operate under conditions in which communication is very unreliable, for example due to electromagnetic

interference or smoke. Moreover, the failure of a limited number of sensors should have little impact on the operation of the remaining functioning sensors. More generally, robustness refers to the ability of the sensors to sustain an acceptable level of performance in the face of unexpected events.

**Autonomy** The sensors should have control over their own actions, without being governed by an external system. The reason for this is that in order to effectively control the sensors such a system requires the observations of all sensors in one place. The amount of communication required to achieve this quickly exceeds the available bandwidth when increasing the number of sensors. Instead, each sensor should be capable of making its own decisions, based on the world picture that it has been able to compile from its own observations and communication with other sensors. In other words, the sensors should be able to determine which actions will achieve better situational awareness. Obviously, autonomy is related to robustness, since autonomy means that there is no central control system that constitutes a single point of failure. Having such a system would make the sensors highly vulnerable to the command link being severed. Moreover, autonomous sensors do not require human attention, which is often a scarce resource in disaster response situations.

**Scalability** It should be possible to scale up the size of a team of sensors in order to adapt to large environments, such as large warehouses or large sections of airspace. This property specifically refers to the ability of the coordination algorithms to handle communication with a growing number of neighbouring sensors.

**Modularity** The composition of the team of sensors should be able to be tailored to the requirements of the mission. Ensuring that different types of sensors can cooperate in a single team necessitates that the coordination algorithm does not impose restrictions on the implementation of the sensors (e.g. whether the control inputs of the sensors are discrete or continuous, the frequency with which they take measurements, or whether the sensors are ground robots or unmanned aerial vehicles).

Thus far, previous work on sensor networks and robotics has fallen short of providing solutions that exhibit all these properties, and thus of providing an integrated approach for monitoring environmental phenomena within dynamic environments with mobile sensors (see Chapter 2 for more details). For instance, one branch of research focuses on the deployment of fixed or mobile sensors under the assumption that the characteristics of the environment are known beforehand, and that the environment can be considered static during the time required to sample from it (Singh et al., 2007; Guestrin et al., 2005; Krause et al., 2006; Zhang and Sukhatme, 2007; Meliou et al., 2007). In more detail, these algorithms are typically geared towards solving a one-shot optimisation problem in an offline phase and are therefore incapable of adapting to changing circumstances.

Moreover, these algorithms do not model the temporal dynamics of spatial phenomena; they consider how relevant phenomena in the environment varies in space, but not in time. As a result, they cannot project the current state of the world into the future, since knowledge of temporal dynamics is a *sine qua non* of prediction, thereby failing to achieve the third state of SA. Finally, these algorithms are centralised. As mentioned before, centralisation puts high demands on the communication network between the sensors, because the observations gathered by the sensors need to be sent to a central control system. More importantly, centralisation creates a single point of failure, thereby increasing the vulnerability of the information stream, which is highly undesirable in hostile environments. As a consequence, they violate the requirement that the sensors should be autonomous, adaptive, and robust. As such, they are less suitable in our domain. We will discuss these approaches in more detail in Section 2.5.1.

In contrast to these methods, a different set of approaches is concerned with the use of adaptive wireless sensors for environmental monitoring, in cases where the characteristics of the environment are not known *a priori* (Kho et al., 2009; Padhy et al., 2007; Osborne et al., 2008). However, this work tends to focus exclusively on static sensors, whose presence is not guaranteed in disaster scenarios. Nevertheless, the techniques employed in these approaches are useful in the context of our work, and will be discussed further in Section 2.5.2.

Finally, there is a significant body of research on mobile robotics for target tracking, event detection and map building (e.g. Grocholsky (2002); Calisi et al. (2007); Murphy et al. (2000); Ahmadi and Stone (2006)). These approaches are robust, modular, and treat each robot as an autonomous decision maker. However, this work does not focus on monitoring environmental parameters, or is exclusively geared towards monitoring one specific type of phenomenon (Kato and Mukai, 2005; Lilienthal et al., 2003). This in undesirable, because we wish to be able to monitor a wide range of environmental phenomena. Nevertheless, in Section 2.5.2, we show the usefulness of the coordination mechanisms employed in these approaches, and how they can be adopted in our work.

In the next section we outline how we will attempt to address these shortcomings in our research.

## 1.1 Research Contributions

The ultimate aim of our research is to provide an integrated set of principled techniques for coordinating multiple sensors for environmental monitoring. To this end, we intend to develop algorithms that allow these sensors to operate in realistic and complex environments. Among these are algorithms that accurately model the environment in a distributed manner, and coordination mechanisms that allow the sensors to exhibit the aforementioned desirable properties.

As a starting point, we have focussed on environments in which mobile sensors need to monitor a single environmental phenomenon. For these environments, we have developed novel solutions that exhibits the properties of accuracy, adaptiveness, robustness, scalability, and autonomy. More specifically, we present the results of the research performed so far: a set of novel online, decentralised coordination algorithms for teams of mobile sensors in which path planning and modelling are simultaneously performed online. To this end, we represent each sensor as an autonomous learning agent. These agents are capable of making measurements, exchanging observations with other agents, modelling both the spatial and temporal characteristics of the phenomena without prior knowledge, and negotiating with their neighbours to reposition themselves in order to improve SA. Moreover, because every agent controls its own movements using information it possesses locally, there is no central point of control, and the coordination mechanism is decentralised.

More specifically the contributions of this report are as follows:

**Un-negotiated Decentralised Coordination (Chapter 4)** — An *un-negotiated coordination* algorithm for online, decentralised path planning, where coordination between sensors is established exclusively through the exchange of observations; sensors need not coordinate (negotiate) about the actions they are about to take, but base their decisions solely on their picture of the state of the environment. This algorithm is based on two important techniques found in previous work. Firstly, Gaussian process regression (Rasmussen and Williams, 2006; Osborne et al., 2008), which is used for processing the raw observations obtained by the sensors and modelling the environmental phenomena. Secondly, myopic information-theoretic control (as found in Grocholsky (2002); Grocholsky et al. (2005)), which is used for maximising the informativeness of the samples that are obtained by moving the sensors to locations where the environment is more uncertain.

**Negotiated Decentralised Coordination (Chapter 5)** — An extension to the previous algorithm, resulting in an *negotiated coordination* algorithm. This algorithm is based on the max-sum message passing algorithm for decentralised control (Farinelli et al., 2008), which allows the sensors to maximise a team objective function in a decentralised way. To make this algorithm computationally feasible to solve the mobile sensor coordination problem, we develop two pruning algorithms that drastically reduce the amount of computation required. These pruning algorithms are generic in the context of the application of max-sum, and are thus not limited to the mobile sensor setting.

**Negotiated Coordination with Continuously Valued Actions (Chapter 6)** — A further extension to the negotiated algorithm for sensors that are characterised by continuous control parameters. For example, sensors that are controlled by setting

a heading and speed, instead of choosing from a finite, discrete set of possible actions. To this end, we develop a continuous version of the max-sum algorithm.

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

1. R. Stranders, A. Rogers, and N. R. Jennings. A Decentralised, Online Coordination Mechanism for Monitoring Spatial Phenomena with Mobile Sensors. In *Proceedings of the Second International Workshop on Agent Technology for Sensor Networks (ATSN)*, Estoril, Portugal, 2008.

2. R. Stranders, A. Farinelli, A. Rogers, and N. R. Jennings. Decentralised Coordination of Mobile Sensors using the Max-Sum Algorithm. In *Proceedings of the 21st International Joint Conference on AI (IJCAI)*, Pasadena, USA, 2009.

3. R. Stranders, A. Farinelli, A. Rogers, and N. R. Jennings. Decentralised Coordination of Continuously Valued Control Parameters using the Max-Sum Algorithm. In *Proceedings of the Eighth International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS-09)*, Budapest, Hungary, 2009.

## 1.2 Report Outline

The remainder of this report is organised as follows:

- In Chapter 2, we discuss related work. The first part of this chapter focuses on the potential application domains in which our research can contribute. In the second part, we review the techniques that have been employed in related work, determine their deficiencies in terms of our requirements, and identify the key techniques that are adopted in our own work.

- In Chapter 3 we present a formalisation of the mobile sensor monitoring problem: we define the layout of the environment in which the sensors operate, their objectives, and the phenomena they monitor.

- In Chapter 4 we describe a *greedy*, *un-negotiated algorithm* for coordinating mobile sensors. This algorithm has the predicate *un-negotiated*, because it achieves coordination between sensors exclusively by sharing observations, and does not require the sensors to communicate about the actions they are about to take. Moreover, it is *greedy*, because sensors maximise their own immediate performance, and ignore the effect of their actions on both the performance of other sensors, and their own future performance. Despite this, however, we show in simulation that this algorithm results in coordinated behaviour, and acceptable team performance.

- In Chapter 5, we extend the algorithm from Chapter 4 to *negotiated coordination* mode, in which sensors negotiate about their joint movements through their environment. In contrast to the un-negotiated algorithm, this algorithm enables sensors to maximise the performance of the entire team, instead of only their individual performance. We show that this results in increased performance compared to the un-negotiated algorithm. The algorithm is based on the max-sum algorithm (Farinelli et al., 2008) that maximises a team objective function through local computation and message passing, and is thus fully decentralised.

- In Chapter 6 we investigate a further extension to the algorithm from 5, which makes it possible to control sensors that are characterised by continuous control parameters, instead of modelling the action space of the sensors as a finite, discrete set of possible moves. To this end, we generalise the discrete max-sum algorithm to the continuous case.

- Finally, in Chapter 7 we present our conclusions and determine the steps that need to be undertaken as part of future work.

# Chapter 2

# Literature Review

In this chapter, we will provide the background against which our research is positioned. In order to do this, we will discuss previous work on sensor networks from two perspectives: its intended applications and the algorithms and techniques that have been used within it. In the first part of the chapter, we focus on the application dimension, where we distinguish between the use of fixed and mobile sensors. We will see that, while fixed sensor networks are suited for different kinds of application scenarios, there is still sufficient similarity with the applications of mobile sensors to warrant an investigation into the usefulness of their techniques for the purposes of our own research. In the second part of the chapter, we highlight the technical dimension, in which we discuss the important concept of adaptive sampling. This involves the maximisation of information gain subject to the sensors' scarce resources, such as communication bandwidth and battery power. More specifically, we will look at related work in terms of the three essential questions that an adaptive sampling algorithm needs to address:

1. Information processing: how are individual samples processed to obtain a high-level representation of the environment? (Section 2.3)
2. Valuing information: how should samples be ranked in terms of their informativeness? (Section 2.4)
3. Maximising information gain: how to control the sensors in such a way that the total information gain of the wireless sensor network is maximised subject to limited battery life and communication bandwidth? (Section 2.5)

Finally, we summarise the discussion and indicate which of the existing areas of research will form the point of departure for our work, and the areas where new work is needed in order to meet the requirements outlined in Chapter 1.

## 2.1 Applications of Wireless Sensor Networks

The protocols and techniques we use in our approach are more broadly applicable than to emergency response alone. In this section, we will therefore discuss the most prominent applications scenarios of wireless sensor networks (WSNs). However, this section is not intended to be an exhaustive overview of all the application domains where wireless sensing technology is used. Instead, we will cover those applications that bear a similarity to our work, or to which our work can contribute. In particular, we will focus on sensing spatial phenomena (such as temperature, humidity and pressure), and on areas related to disaster management (such as military surveillance and mapping of unknown environments). The section is split up into three parts. In the first, we review the literature on applications of *fixed* wireless sensor networks. Next, we shift our attention to robot-mounted or *mobile* sensors. Finally, we look at *hybrid* approaches, that use both fixed and mobile sensors. This is a natural distinction to make, because the properties of the applications of fixed sensor networks are inherently different from those in which mobile sensors are used. Nevertheless, we will explain why we believe that mobile sensors could—under certain circumstances—prove to be worthwhile in the application domains of fixed sensor networks, and how presence of a fixed WSNs can be exploited by a mobile sensor network.

### 2.1.1 Fixed Wireless Sensor Networks

Fixed WSNs consist of small, energy efficient and relatively cheap sensor nodes. These nodes can sense a wide range of phenomena, from light intensity and temperature, to biological agents and gas concentration. They can be easily deployed at low cost without the need of an existing (wired) communication infrastructure, can perform their tasks largely unsupervised, and can provide fine grained monitoring. These properties have led to their application in a wide range of domains.

For example, WSNs have been successfully exploited in agriculture, climate change research, military surveillance, and environmental control. In more detail, the application of wireless sensing in agriculture has led to better insights into various agricultural processes on a micro-scale (Zhang, 2004; Edordu and Sacks, 2006; Galmes, 2006; Langendoen et al., 2006; Wark et al., 2007) (see Figure 2.1(c)). In crop cultivation, for example, the availability of detailed and localised information about humidity, temperature and sunlight allows farmers to administer insecticides more effectively, to irrigate crops more evenly, prevent the spreading of diseases, and better predict crop yields, thus saving time, money, and environmental damage.

Related to agriculture, environmental monitoring has also seen the successful application of WSNs. Here, WSNs are used to deliver data about highly localised environmental phenomena to ecologists, climatologists and geologists. For instance, WSNs have been

used in climatic change studies, for example by monitoring the climate on a micro-scale, such as the dynamics of glaciers over a long period of time (Padhy et al., 2007; Hart et al., 2006) (see Figure 2.1(a)), monitoring and detecting floods (Kho et al., 2009) (Figure 2.1(a)), monitoring animals and the micro-climates within their habitats (Szewczyk et al., 2004), and local weather and tide predictions (Osborne et al., 2008). For a general overview of the use and benefits of sensor networks for environmental monitoring, refer to Hart and Martinez (2006) and Rogers et al. (2009).

On an even smaller scale, wireless sensor networks have been applied in environmental control of "intelligent buildings". At the Intel Berkeley Research Lab, for example, a network of sensor nodes has been deployed that measures the light intensity, humidity, and temperature within the building at regular intervals.[1] This data can be subsequently used as input for sophisticated control systems for heating, ventilation, air conditioning (HVAC) that allow fine-grained adjustments of the building's climate (Deshpande et al., 2005a). In doing so, the consumption of electricity can be reduced, with the same or higher comfort level for the building's inhabitants.

The use of fixed wireless sensors has also been suggested for detecting moving targets in military and security domains (e.g. Dash et al., 2005; Vargas et al., 2003). In such cases, the nodes of the WSNs are equipped with range and bearing sensors that allow them to localise a moving target. Measurements from multiple sensors are then combined (fused) together to improve the accuracy of the resulting position estimate.

Finally, in disaster response, WSNs are used in various ways. For example, wireless sensors can be used to gather data as input for flood prediction models (Mandal et al., 2005), localise victims after a building collapse by triangulating mobile transmissions emanating from their location (e.g., from mobile phones), detect seismic activity, and determine the structural integrity of a building (Fujiwara et al., 2004; Kurata et al., 2006). There is also a branch of research that investigates the use of wireless sensor networks to improve situational awareness to emergency services. For example, Sha et al. (2006) propose a wireless sensor network architecture, in which firefighters are equipped with wireless sensing equipment. The sensors continuously track the location of the firefighters, monitor their actions and keep track of the environmental conditions within the disaster area. The obtained data is then transferred to a command centre, where centralised resource allocation and scheduling take place. With similar requirements in mind, Klapwijk (2005) developed a distributed communication and task allocation architecture for assisting firefighters with collecting and disseminating relevant information through a network of devices that are carried by emergency response personnel. Finally, we would like to refer the interested reader to an entertaining perspective on the future potential of WSNs in disaster response and prevention by Elson and Estrin (2004).

[1] See `http://db.csail.mit.edu/labdata/labdata.html`

(a) A sensor node for detecting floods (source: `http://envisense.org/floodnet/deployment.htm`)

(b) The Briksdalsbreen glacier in Norway that is monitored by a large-scale WSN (source: `http://envisense.org/glacsweb/photos/briks-panos/index.html`

(c) A sensor for measuring moisture levels in soil (source: `http://images.businessweek.com/ss/08/12/1205_sb_drought/8.htm`)

FIGURE 2.1: Examples of deployment scenarios for WSNs

In this section, we have reviewed the applications of WSNs, with an emphasis on environmental monitoring and disaster response. The fact that these WSNs serve multiple purposes in this area shows the relevance of fixed WSNs in our intended application area, and a range of application scenarios for our own work. However, it cannot always be assumed that a fixed WSN is present in the scenarios where they could play a vital role. In such cases, the use of mobile sensors could present a more fitting solution. This will be discussed in the next section.

### 2.1.2 Mobile Wireless Sensor Networks

Mobile wireless sensor networks consist of sensors mounted on mobile autonomous robots. These robots can range from mobile ground robots, to small helicopters or fixed wing aircraft; where the latter two are usually referred as Unmanned Aerial Vehicles (UAVs) (for examples, see Figure 1.1). The benefits of having mobile sensors in patrolling in a dynamic environment are clear: they can be the eyes and ears of decision makers, and provide them with the tools to make more informed decisions. In a sense, they perform the tasks of a fixed WSN where none is available. More importantly, by patrolling the area and taking measurements at different locations, a small number of mobile sensors are able to achieve an equivalent spatial resolution to a large fixed WSN. Moreover, they can autonomously deploy themselves in a timely manner, which is very desirable property when time is of the essence, such as in disaster response.

For example, in reconnaissance and surveillance, mobile robots equipped with sensors are used for target tracking, exploration, and localisation. In particular, Grocholsky (2002) and Grocholsky et al. (2003, 2006) have shown the effectiveness of using teams of mobile robots for tracking targets. These teams consist of autonomous ground robots and unmanned aerial vehicles, that coordinate their actions to improve tracking accuracy. Within this context, Makarenko and Durrant-Whyte (2006) present a generic framework for the overarching task of information gathering. This framework allows different types of robots, equipped with various kinds of sensors, to work together to collect information about multiple entities within the environment. They show how the framework can be instantiated for the task of multi-robot target tracking and demonstrate its effectiveness in an example tracking scenario. Ahmadi and Stone (2006) also develop an approach for multi-robot patrolling to maximise event detection, such as intrusions or fires. Finally, Agmon et al. (2008) present an centralised control algorithm for multiple sensors for perimeter control in the presence of different types of adversaries. Their problem formulation is interesting, because unlike our model, it assumes that the environment reacts strategically to the actions of the sensors. In future work, we intend to adapt the algorithms developed in this report to provide a decentralised solution to this problem.

Somewhat similar to patrolling, simultaneous localisation and mapping (SLAM) is another application of mobile robotics. In the intended scenarios, mobile robots equipped

with laser range finders are introduced into an uncharted environment. Working in cooperation, the robots create a map of the environment, while at the same time keeping track of their location relative to the environment (Stachniss et al., 2005). In a similar vein, Roy et al. (1999) use a mobile robot as a tour guide in a museum, that, in order to perform its task, must avoid the visitors, and keep track of its location. Mobile robots have also been used for patrolling through an environment, and detecting potential intruders (Grace and Baillieul, 2005). Clearly, these techniques are relevant to our work, since mobile sensors in dynamic environments are often not only uncertain about the environmental variables within them, but are also faced with uncertainty about the physical layout of their environment.

In disaster management itself, which is our running example of these dynamic environments, the use of mobile robots can reduce the need for exposing humans to hostile environments. For instance, mobile robots can be employed to trace the source of a gas leak (Kato and Mukai, 2005; Lilienthal et al., 2003), and on a longer term, researchers even intend to perform search and rescue tasks with robots (Calisi et al., 2007; Murphy et al., 2000).

Clearly, in all these previous examples, mobility is a requirement of the tasks the sensors are performing. In all of the aforementioned applications, no existing fixed wireless sensor network is available, and the inherent time constraints, uncertainty and possible hostility associated with the environment make a manual deployment of a WSN impossible. However, the use of mobile sensors is not limited to dynamic and time constrained scenarios such as disaster management. Their use has also been suggested in areas that are too large to be able to obtain sufficient resolution with fixed sensor nodes in a cost-effective manner, or where the terrain makes deployment of mobile sensors easier than the deployment of fixed WSNs.

A good example that combines the former and the latter is the Networked Infomechanical System (NIMS) (Rahimi et al., 2004; Pon et al., 2005a,b). NIMS is a single sensor suspended on a wire, that can move horizontally and vertically to sense within a vertical plane. It has been used to measure wind speeds and other micro-climate dynamics in forests, and take pH measurements in a cross-section of a river (Krause and Guestrin, 2007). In a similar fashion, autonomous sensing boats have been used to collect measurements to monitor a wide range of pertinent biological phenomena (Dhariwal et al., 2006; Zhang and Sukhatme, 2007; Singh et al., 2007; Meliou et al., 2007). The main benefit of both NIMS and these boats is that they can provide a theoretically infinite spatial resolution even with only a single sensor. Moreover, their deployment in these environments is much less cumbersome than that of a WSN with similar sensing resolution (specifically in a body of water). However, as we will see in Section 2.5, controlling these sensors such that they achieve the desired resolution, particularly in a decentralised fashion, is far from trivial.

### 2.1.3 Hybrid Approaches

In some application domains, mobile sensing robots can benefit from the presence of a fixed WSN, and *vice versa*. To illustrate this within the context of the example given in the introduction, suppose that a sophisticated environmental control system has been installed in the building that uses a WSN to monitor the environmental conditions. Mobile robots deployed within the building can connect to this WSN to receive its observations. As a result, the coverage and spatial resolution obtained by the collection of sensors is increased.

Zhang et al. (2005) propose a solution for city-wide monitoring based on this idea. It involves a combination of fixed and wireless sensor nodes for creating a city-wide sensor network to detect nuclear, biological and chemical attacks. In order to enhance resolution, their solution aims to complement a network of fixed sensors (which are already present in some cities) with sensors that are mounted on taxis, ambulances, and autonomous helicopters, or that are worn by patrolling police officers. Using wireless LANs that are available throughout the city, the sensors periodically send their data to a command centre, from which further action can be taken.

The converse, in which mobile robots are utilised to enhance fixed wireless sensor networks, has also been studied in the literature. For example, Pereira et al. (2004) propose the use of a mobile robot for collecting measurements from sensors with a very limited communication range. This robot visits each sensor in turn, downloading the measurements from the sensors to its own memory for later use. LaMarca et al. (2002) also use a mobile robot to provide various services to a WSN, such as recharging their batteries, redeploying them based on the changed requirements of the environment, and recalibrating them. This last service is especially important in WSNs, because without frequent calibration, the measurement error of the nodes will increase (or drift), making their measurements increasingly unreliable and even unusable. However, manual calibration is often found to be a labour-intensive task, which the use of this mobile robot can alleviate.

We can relax the requirement that mobile robots are purely used as "maintenance men" for redeploying fixed WSNs, as proposed by LaMarca et al., by equipping the robots themselves with sensors, and allowing them to build their own fixed WSN. This WSN can ensure a more permanent coverage of important areas of the environment, while the mobile robots continue their exploration. A similar idea is exploited by Nguyen et al. (2004), who propose the use of small wireless relay nodes that are deployed by a mobile robot. These relay nodes are used to maintain a command link between the mobile robot and a base station outside the environment. Similarly, augmenting these nodes with the capability of sensing their environment could improve sensing resolution as well as communication reliability. We believe this could be an interesting extension that deserves further investigation in future work.

This concludes our overview of the applications of WSNs. There are a few important conclusions that should be drawn from this discussion. First of all, WSNs serve in a wide spectrum of applications within environmental monitoring, demonstrating the relevance of our field of interest. Second of all, and, more importantly, mobile sensors can play a vital role in time constrained scenarios where no fixed WSN is available, or where the environment is simply too large to be effectively covered by a permanent deployment of a fixed WSN.

In light of the requirements laid down in the introduction, mobile WSNs are decidedly more adaptive and robust than their fixed counterparts. Especially in hazardous and dynamic environments, mobile WSNs can increase situational awareness, where the deployment of fixed sensors networks would not be possible or would be too dangerous.

## 2.2 Adaptive Sampling

In the previous section, we focused on the applications of WSNs. In other words, we reviewed *what* these networks are used for. In the second part of this chapter, we move on to *how* these WSNs perform their tasks. Thus, we review several technical aspects of wireless sensor networks, and analyse them from the perspective of the requirements in Chapter 1.

As explained in Chapter 1, the research proposed in this work falls within the field of adaptive sampling (Kho et al., 2009; Zhang and Sukhatme, 2007; Zhou et al., 2006). Informally, adaptive sampling can be thought of as "intelligent sampling". Generally speaking, it is not sufficient to simply deploy a set of sensors and let them sample at a fixed rate. In most dynamic environments, the rate of change varies with time. In such cases, fixed rate sampling will allocate an equal amount of samples to the intervals in which the environment is changing rapidly, as to the intervals in which the environment is fairly static. In so doing, the number of informative samples that can be collected is suboptimal. So, instead, each sensor needs to carefully decide when and where to sample in order to best allocate its resources. Given this, we define adaptive sampling as:

**Definition 2.1** (Adaptive Sampling)**.** Selecting observations in space, time or both, as to maximise information gain by the entire WSN, subject to movement constraints and scarce resources such as communication bandwidth and battery power.

In the introduction, we decomposed the problem of adaptive sampling into three sub-challenges: information processing, valuing information, and maximising information gain. Figure 2.2 shows how these three sub-challenges are addressed by different components inside a sensor node:

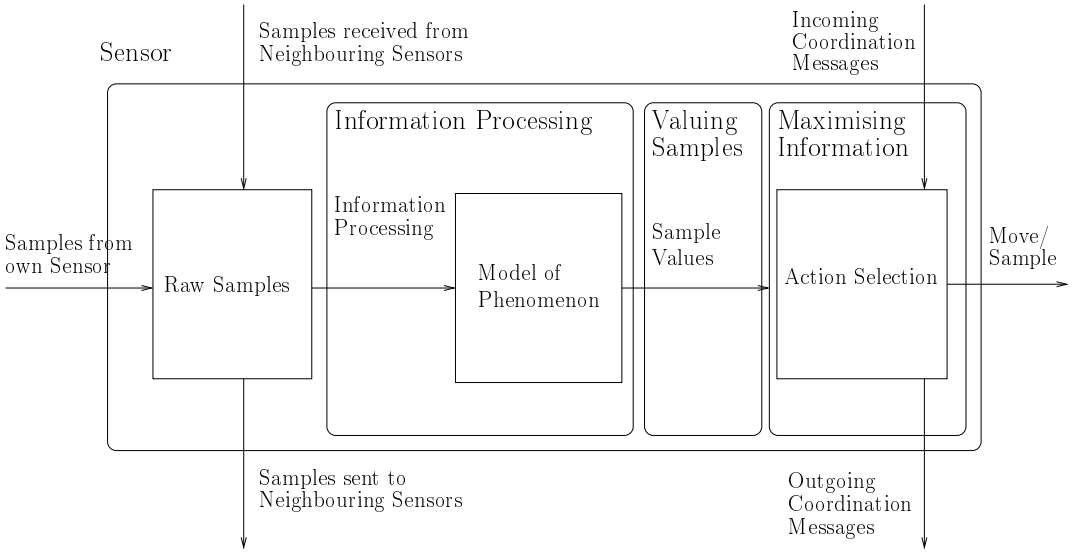



FIGURE 2.2: A general architecture for a sensor node.

1. Information processing. Samples collected by the sensor, either through sensing or communication with others, are processed into a model of the phenomenon.

2. Valuing samples. Samples that are considered for collection are valued (ranked) in terms of their potential to increase situational awareness.

3. Maximising information gain. The sensor decides which action to take next to maximise the value of collected samples. Here, the sensor is not necessarily concerned with maximising the value of the sample it collects itself, but rather with maximising the values of those samples that the team decides to collect. To identify the optimal action, the sensor uses the value of potential samples, and (optionally) negotiation with other sensors. The question of whether or not the sensors should negotiate (i.e. explicitly coordinate) and the challenges involved with such negotiation are the subject of Section 2.5.3.

The next three sections highlight each component in turn. However, in the remainder of this section we classify related work based on the dimension along which is sampled. In more detail, algorithms can take samples over time (temporal sampling), space (spatial sampling), or both (spatio-temporal sampling). The dimensions along which samples are taken determines to a great extent the techniques that are used, and it is therefore important to make this distinction. Moreover, as we have discussed in the introduction, the degree of situational awareness is highly dependent on the availability of both spatial and, especially, the temporal dynamics. Specifically, in order to achieve the third stage of situational awareness—prediction—the temporal dynamics are required to project the current state of the world into the future. Therefore, a good model of the temporal dynamics of phenomena is of vital importance.

### 2.2.1 Spatial Adaptive Sampling

Generally speaking, spatial adaptive sampling includes two distinct sets of approaches. The first consists of algorithms for calculating informative placements for fixed sensors. The second contains approaches that use mobile robots to sample from a static environment, or environments that are changing so slowly that they can be considered static whilst the samples are collected.

In the first set of approaches, concerning fixed sensor placement, the challenge of calculating informative locations has been thoroughly researched by Guestrin et al. (2005). In this approach, the spatial correlations within an environment are assumed to be known. These correlations describe the relations between measurements taken in space, and determine the way in which the environment varies over space. For example, the spatial correlations between temperature measurements in a pond will be smaller on the surface, where the water is exposed to the air, than at the bottom. Now, the algorithm by Guestrin et al. uses correlations that are learnt during an initial deployment of a large number of sensors. Next, their algorithm selects an informative subset of these sensors, after which the rest are removed to reduce cost. Krause and Guestrin (2005) study several budget-restricted cases of the problem, in which a certain cost is associated with the deployment of a sensor. In particular, Krause et al. (2006) extend this work by taking communication cost into account, thereby making an explicit trade-off between the informativeness of a potential sensor location, and the expected cost of sending measurements to a base station.

Now, while these techniques fail to give proper attention to the temporal dynamics of the environment from which they sample, the challenge of placing fixed sensors in space is similar to positioning mobile sensors in space-time, without considering the movement constraints of the sensors. In fact, in Chapter 4, we adapt these sensor placement algorithms for controlling the mobile sensors' movements.

The second set of approaches tackles the challenge of sampling static environments with *mobile* robots. Since these environments change very little or not at all, the temporal correlations between measurements can be ignored. For instance, Rahimi et al. (2004); Pon et al. (2005a,b) and Krause and Guestrin (2007) have used the NIMS architecture (see Section 2.1.2) to collect samples in forests and rivers. Since the environmental conditions within these environments vary slowly relative to the time needed to take the required samples, the environment is considered to be static during sampling. Along the same lines, Singh et al. (2007) and Zhang and Sukhatme (2007) use robotic boats to sample in bodies of water with prior knowledge of the spatial correlations of those environments. As we will see later in Section 2.5.2, this knowledge allows them to precompute paths along which informative samples are likely to be taken.

Since the environment is not changing, or negligibly so, all these approaches attempt to avoid visiting the same (general) location twice, because no new information can be obtained by doing so. It is important to note that this is not necessarily a good strategy in the dynamic environments we are concerned with in our research, since the conditions within these environments are likely to change after a sensor has moved through it. Consequently, in our work, we do not prevent sensors from revisiting previously sampled locations; on the contrary, we explicitly encourage sensors to do this once sufficient time has passed and the environment is expected to have been changed significantly. These considerations lead to the necessity of taking into account the temporal dimension of sampling.

### 2.2.2 Temporal Adaptive Sampling

As suggested before, many environments *do* change over time, and thus the assumption that the environment is static does not hold. This is either caused by the fact that sensors are deployed for a longer time in moderately changing environments, or in environments where the phenomena are changing at a high rate. Examples of the former case are found in environmental monitoring when the mission time is sufficiently long to detect the effect of the day-night cycles or, on a longer time-scale, a seasonal change. Notable examples of the latter case include military surveillance and disaster scenarios. However, in both cases, samples must be carefully scheduled over time in order to sample efficiently from them; sampling at a constant rate wastes resources in periods where the relevant variables are changing slowly, and result in less-than-optimal sampling resolution at times when they are changing rapidly.

In tide prediction, for example, Kho et al. (2009) take this rate of change within the environment into account when creating a sampling schedule at the beginning of the day. Their planning algorithm considers the amount of energy left in the sensor's battery, and the amount of information that can be collected at different times of the day, based on past experience. At intervals when the tide is expected to come in, the water height will change quickly, and the sampling frequency is increased. In contrast, Osborne et al. (2008) do not explicitly schedule sampling times, but instead only decide when the next sample should to be taken. This decision is based on the rate of change that has been detected from samples taken by both the sensor itself, and the other sensors with which it communicates. Padhy et al. (2007) take a similar approach, and have each sensor schedule a new measurement at times when its information value is expected to exceed the average information value of the last few measurements.

The key lesson that should be learnt from these approaches, is that the sample rate should be adjusted to the rate at which the environment is changing. Translated to a mobile sensor setting, this implies that the robots should take more samples when or

where the environment is changing rapidly (for example in the parts of a building that are on fire), and take less samples when or where it is not.

### 2.2.3 Spatio-temporal Adaptive Sampling

Thus far, we have considered sensors that model either the spatial or temporal dynamics of their environment, but not both. However, recall the definition of situational awareness given at the start of the Chapter 1. This definition stresses the importance of perceiving elements in space *and* time. So, unlike the applications of mobile sensors discussed in Section 2.2.1, it is not always justified or prudent to consider the environment to be static during the time it takes for sensors to sample from it. This is either caused by the fact that an environment is highly dynamic, and thus changes rapidly, or by the fact that the sensors can only cover cover a small fraction of the environment at a given time, and consequently require a long time to traverse it. In the latter case, even if the environment changes slowly, conditions might have changed significantly by the time the sensors have performed a single sweep through it. In other words, in these environments, time is a variable that simply cannot be neglected. This means that mobile sensors need to be able to determine how to sample in space-time in order to be able to reconstruct the dynamics of the underlying phenomenon. As a result, it might be necessary to revisit locations in order to keep track of these dynamics.

Our contributions presented in this report fall within this category of spatio-temporal sampling. The disaster scenarios that provide the motivating examples for this work are characterised by both uncertainty and dynamism. Consequently, the sensors are aware of the fact that the environment around them is changing and thus they need to keep track of these changes. We will discuss the way in which these dynamics are modelled and how this influences the scheduling of future sampling locations in Sections 2.3 and 2.5.

Apart from our own work, we are aware of the application of spatio-temporal sampling in gas source tracking (Kato and Mukai, 2005; Lilienthal et al., 2003; Zhang et al., 2005) and target tracking (Grocholsky, 2002; Grocholsky et al., 2006). Whereas these approaches are very relevant to our own objectives, and our work is even inspired by them, they are fairly specific to their application domain. We intend to develop techniques for monitoring and reconstruction environmental phenomena in general, and not focus on just a single one. Therefore, the work by Osborne et al. (2008) is highly relevant to our work, because they model both the spatial covariance between measurement taken by multiple fixed weather sensors as well as the temporal covariance that exists between measurements taken by a single sensor. In order to do so, they utilise the versatile Gaussian process (GP) (see Section 2.3.2), which allows for a wide range of phenomena to be modelled. We come back to the GP in Section 2.3.2.

Meliou et al. (2007) approach the problem from a more generic point of view, in which no assumptions are made about the type of the phenomenon. In their work, time is discretised into time slices (or epochs), during which the phenomenon is assumed static. Samples gathered during previous time slices are used to improve prediction accuracy for current and future time slices. However, for the assumption to hold that the phenomenon is static during a single time slice in very dynamic environments, the length of each time slice has to be short. As a result, mobile sensors need to traverse the environment very quickly to prevent low prediction quality. Clearly, this is not always possible. This is especially true since the authors impose the requirement that the robots need to return to a base station at the end of every time slice. While this approach is attractive and bears similarity to our own, we believe that this assumption is too strong.

To sum up, in order to achieve situational awareness in a dynamic environment, we strongly believe that neither the temporal, nor the spatial dynamics that govern the monitored phenomena can be neglected. On the one hand, focusing solely on the spatial correlations might give an accurate picture of the current state of the world, but not of what will happen in the near future. On the other hand, being able to predict what will happen in only a small area of the environment results in an incomplete spatial picture of the world. Clearly, making decisions based on either spatially or temporally deficient situational awareness exposes the decision maker to risks.

## 2.3 Information Processing

In the previous section, we offered a classification of existing work in terms the types of dynamics that are taken into consideration by different methods of adaptive sampling. However, the question of *how* sampling points are selected still remains unaddressed. In the introduction, we decomposed the problem of adaptive sampling into three components: information processing, valuing information, and information maximisation (see Figure 2.2). In this section, we will address the first component—information processing—by looking at how the obtained samples are aggregated and transformed into a high-level representation of the phenomenon.

Specifically, we will first focus on the use of regression to detect patterns in the raw sensor measurements. As discussed in the previous section, these patterns are required to predict (or extrapolate) the future state of the world. More specifically, we discuss the two types of regression that are commonly found in adaptive sensing: linear regression, and the more powerful Gaussian process regression. The latter will receive some more in-depth coverage, because of the relevance to our own work. Finally, we discuss techniques that do not fall in either category, but which are, nevertheless, relevant to our work.

### 2.3.1 Linear Regression

Linear regression models a phenomenon by a collection of linear relations between the explanatory variables (such as time and location) and the variable of interest (such as temperature or gas concentration). Its attractiveness stems from its simplicity and computational efficiency. For this reason, Kho et al. (2009) employ linear regression to obtain a piecewise linear approximation of the relation between time and water height in their flood monitoring system. Linear regression is also employed in environmental monitoring by Padhy et al. (2007). However, the environmental variables in their setting (i.e. temperature and pressure) are characterised by their non-linear relation with time. Consequently, Padhy et al. model these variables with piecewise linear functions, whereby Bayesian inference is used to decide whether the newly obtained sample can be sufficiently explained by the current regression model, or whether the sample represents a change point and the model needs to be discarded in favour of a new one. In the latter case, a new line segment is added to model the phenomenon. In the two-dimensional (spatial) case, Zhang and Sukhatme (2007) employ linear regression to model temperature variations in a body of water, where temperature measurements are expected to be inversely correlated to the distance between the locations at which they were taken. As we will see, this is equivalent to using a Gaussian process with a linear regression covariance function (Rasmussen and Williams, 2006, Sections 2.1.1 and 4.2.2).

Although linear regression is attractive due to its simplicity and its low computational cost, it lacks universal applicability; many real-life phenomena are governed by (strongly) non-linear relations. Moreover, in light of the requirements defined in Chapter 1, we do not wish a regression method to restrict the class of phenomena our research is applicable to, even if this results in less efficient algorithms. That said, we believe that we should allow for a more flexible type of regression, and, if necessary, re-evaluate our choice at a later stage. With this in mind, we will see that the Gaussian process described in the next section provides a more suitable alternative.

### 2.3.2 Gaussian Processes

Gaussian processes (GPs) provide a more flexible way of performing regression than linear regression, because they are capable of performing regression over large classes of functions, and are therefore not limited to (piecewise) linear functions alone. In this section we will offer a basic introduction to the Gaussian process. First, we will explain its basic properties, and describe the important role played by the covariance function. Then, in Section 2.3.2.1, we explain how a GP is fitted to observations made by a sensor when the parameters of the covariance function are known. Next, in Section 2.3.2.2 we relax this assumption, and discuss techniques that are capable of inferring these parameters from data. Finally, in Section 2.3.2.3, we discuss the use of the GP in related work, and we explain how its adoption impacts our own work.

Before we continue with the specifics of the GP, however, we will define some of the notation that will be used throughout this report. Let the process $y = f(\mathbf{x})$ denote the relation between a $D$-dimensional input-vector $\mathbf{x} \in \mathbb{R}^D$, and an output variable $y \in \mathbb{R}$. Moreover, let $\{(\mathbf{x}_i, y_i)|i = 1 \dots n\}$ denote a set of input-output pairs, which represent the observations of this process $f$. This set is commonly referred to as the training set. Furthermore, variables subscripted with a $*$ (such as $\mathbf{x}_*$) relate to predictions (or test data): $y_*$ denotes the predicted function value for (possibly unobserved) input-vector $\mathbf{x}_*$. Finally, we denote the $n$-dimensional vector of all collected outputs $y_i$ as $\mathbf{y}$, and collect the $n$ inputs $\mathbf{x}_i$ in a $D \times n$ matrix $X$.

Now, Rasmussen and Williams (2006) define the GP as:

**Definition 2.2** (Gaussian process)**.** A Gaussian process is a collection of random variables, any finite number of which have a joint Gaussian distribution.

It is well-known that a finite set of jointly Gaussian random variables is fully determined by their mean $\boldsymbol{\mu}$ and covariance matrix $\Sigma$.[2] Now, given this fact, we can generalise these definitions of mean and (co)variance to an infinite set of variables, and fully determine a Gaussian process by a mean function $m(\mathbf{x})$ and covariance function $k(\mathbf{x}, \mathbf{x}')$:

$$\begin{aligned} m(\mathbf{x}) &= E[f(\mathbf{x})] \\ k(\mathbf{x}, \mathbf{x}') &= E[(f(\mathbf{x}) - m(\mathbf{x}))(f(\mathbf{x}') - m(\mathbf{x}'))] \end{aligned}$$

where $E[X]$ is the expectation of random variable $X$. Note that the covariance function $k$ determines the covariance between two outputs of $f$ as a function of their associated inputs $\mathbf{x}$ and $\mathbf{x}'$. Now, if both $m(\mathbf{x})$ and $k(\mathbf{x}, \mathbf{x}')$ are given, we have a prior over the function $f$. This prior defines a probability distribution over possible functions, when no knowledge about input-output pairs of the function are available. However, if we also have access to input-output pairs $\{(\mathbf{x}_i, y_i)|i = 1 \dots n\}$ of the function $f$ (the training data), we are able to incorporate this additional knowledge into the Gaussian process, and improve the accuracy of predictions about the function values $\mathbf{y}_*$ (the test data) at unobserved locations. Here, we exploit the fact that the prior joint distribution of $[\mathbf{y}, \mathbf{y}_*]^\top$ is multivariate Gaussian:

$$\begin{bmatrix} \mathbf{y} \\ \mathbf{y}_* \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} K(X, X) & K(X, X_*) \\ K(X_*, X) & K(X_*, X_*) \end{bmatrix}\right)$$

where the matrices $K(X, Y)$ are obtained by evaluating the covariance function $k$ for all pairs of columns from the matrices $X$ and $Y$. So, $K(X, X)$ is the covariance matrix for all pairs of training points, and the matrix $K(X, X_*)$ contains entries that specify the

[2] Refer to Appendix A.1 for the mathematical background on the Gaussian distribution that is used throughout this section.

correlation between all pairs of training and test points. The posterior distribution of $\mathbf{y}_*$ given $\mathbf{y}$ can now be easily obtained from Bayes' theorem, using the properties of the Gaussian distribution (see Appendix A):

$$P(\mathbf{y}_*|X_*, X, \mathbf{y}) = \mathcal{N}(\boldsymbol{\mu}, \Sigma)$$

where mean vector $\boldsymbol{\mu}$, and covariance matrix $\Sigma$ are obtained by:

$$\boldsymbol{\mu} = K(X_*, X)K(X, X)^{-1}\mathbf{y} \tag{2.1}$$

$$\Sigma = K(X_*, X_*) - K(X_*, X)K(X, X)^{-1}K(X, X_*) \tag{2.2}$$

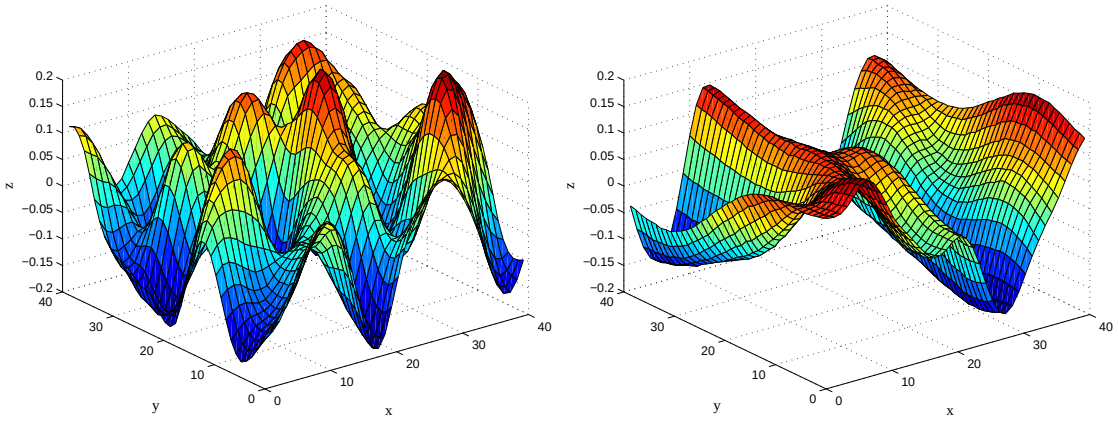
Note that the predictive mean of $\mathbf{y}_*$ is a linear combination of training inputs $\mathbf{y}$, and its covariance has been decreased. Also, it is important to note that the covariance of $\mathbf{y}_*$ does not depend on the actual observations $\mathbf{y}$, but only on their input vectors $X$. Put differently, if the covariance function is known, the covariance of the predictions depends only on the locations where the samples $\mathbf{y}$ have been obtained, not on the values of the samples themselves. As we will see later, this is a property of GPs that is exploited by algorithms that calculate informative placements of fixed sensors, without the need for actual sampling.

In the Gaussian process, the covariance function $k(\cdot, \cdot)$ plays a critical role; it determines the covariance between two sample points of the process $f$ and thus it restricts the class of functions over which the Gaussian process performs regression. To demonstrate this with an example, and show the versatility of the Gaussian process, a simple version of the local linear regression model discussed earlier can be obtained by using the covariance function:

$$k(\mathbf{x}, \mathbf{x}') = \sigma_0^2 + \mathbf{x} \cdot \mathbf{x}' \tag{2.3}$$

This corresponds to putting a $\mathcal{N}(0, 1)$ prior on the weights of the elements of $\mathbf{x}$, and a prior of $\mathcal{N}(0, \sigma_0^2)$ on the intercept of the linear function (Rasmussen and Williams, 2006, Section 4.1). Put differently, local (linear) regression is a special case of the Gaussian process.

However, for non-linear relations, the squared exponential covariance function is a more typical choice; it models the correlations between two sample points for a large class of smooth, non-linear functions. Here, the covariance between outputs is inversely proportional to the distance between their corresponding inputs. In its simplest form, the function is defined as:




(a) A function drawn from a process with equal length-scales along both dimensions.

(b) A function drawn from a process with a longer length-scale along the y dimension than along the x dimension. Note that the process now varies less rapidly along the y dimension.

FIGURE 2.3: Two bi-variate functions drawn from GPs with a squared exponential covariance function, showing the effect of varying the lengthscales.

$$k(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{1}{2}\left|\mathbf{x} - \mathbf{x}'\right|^2\right) \tag{2.4}$$

The use of this covariance function leads to very smooth processes. This is due to the fact that it is infinitely differentiable, resulting in infinitely mean-square differentiability of the process (Rasmussen and Williams, 2006, Section 4.2.1). Moreover, the process will exhibit the same characteristics over each of its input dimensions. That is, the process is insensitive to translation and rotation. Such a process is called *stationary*. For many processes encountered in reality this might not apply, in which case a transformation matrix $\mathbf{P} = diag(l_1^2, \ldots, l_D^2)$ is introduced, and $|\mathbf{x} - \mathbf{x}'|$ is replaced by $(\mathbf{x} - \mathbf{x}')^\top \mathbf{P}^{-1}(\mathbf{x} - \mathbf{x}')$. The entries $l_1^2, \ldots, l_D^2$, scale the dimensions of the input vector $\mathbf{x}$ independently.[3] Depending on the type of dimension to which they apply, these entries are more commonly referred to as *length-scales* or *time-scales*. Generally, the more gradually the modelled phenomenon varies over an input dimension, the longer length-scale for that dimension, and *vice versa*. To put this in the context of sensor networks, the individual scales allow us to model processes that are strongly correlated along one input dimension, while weakly correlated along another. To illustrate this, Figure 2.3 shows an example of the effect of varying the lengthscales in a two-dimensional process.

A third often-encountered class of covariance functions is the Matérn. Whereas the use of Squared Exponential leads to infinitely mean-square differentiability of the process, the Matérn class makes it possible to explicitly control the differentiability of the process

[3] Note that for $l_i = 1$ for $1 \leq i \leq D$, the numerator of the exponent becomes the square of the Euclidean distance between the two vectors, in which case the resulting GP exhibits the same characteristics over all input dimensions.

through the variable $v$. In fact, the process is $k$ times mean-square differentiable iff $v > k$:

$$k(r) = \sigma_f^2 \frac{2^{1-v}}{\Gamma(v)} \left(\sqrt{2v}r\right)^v K_v \left(\sqrt{2v}r\right) \tag{2.5}$$

where $K_v$ is a modified Bessel function, $\sigma_f^2$ is the signal variance, and $r$ is defined as:

$$r \equiv \sqrt{(\mathbf{x} - \mathbf{x}')^\top \mathbf{P}^{-1} (\mathbf{x} - \mathbf{x}')} \tag{2.6}$$

Here, $\mathbf{P}$ is a diagonal matrix with lengthscales, as defined before. Now, for $v = 3/2$, Equation 2.5 reduces to:

$$k(\mathbf{x}, \mathbf{x}') = \sigma_f^2 \left(1 + \sqrt{3}r\right) \exp\left(-\sqrt{3}r\right) \tag{2.7}$$

The Matérn class in general, and the special case in Equation 2.7 in particular, are found to be very suitable for modelling less smooth functions that are commonly found in environmental monitoring (Stein, 1999). More specifically, the Matérn is able to accurately model certain aspects of environmental phenomena, making it attractive from the perspective of the requirements laid down in the Chapter 1. We will therefore encounter it again in Chapter 4, where we introduce our own work.

Thus far, we have assumed noise-free observations. However, in many practical sensor deployments observation noise cannot be neglected. In this case, the sensors do not observe the process $f(\mathbf{x})$ itself, but a noisy version of it: $f(\mathbf{x}) + \varepsilon$. Assuming that $\varepsilon$ has a Gaussian distribution with variance $\sigma_n^2$ and is independent of the input $\mathbf{x}$, we add an extra term $\sigma_n \delta_{\mathbf{x}\mathbf{x}'}$ to the covariance function, where $\delta_{\mathbf{x}\mathbf{x}'}$ is the Kronecker delta which equals one iff $\mathbf{x} = \mathbf{x}'$. To see why this models noise, note that this term adds $\sigma_n^2$ to the diagonal of the covariance matrix, effectively increasing the variance of the associated output-variable. For example, extending the Matérn covariance function in Equation 2.7 for noisy observations, we obtain:

$$k(\mathbf{x}, \mathbf{x}') = \sigma_f^2 \left(1 + \sqrt{3}r\right) \exp\left(-\sqrt{3}r\right) + \sigma_n^2 \delta_{\mathbf{x}\mathbf{x}'} \tag{2.8}$$

Now, in Section 2.2.3 we argued that modelling both both spatial and temporal correlations of a phenomenon is crucial for achieving good situational awareness. The GP allows us to create processes that have different correlation structures along different dimensions. In its most general form, a process with two spatial and one temporal dimension, where input vector $\mathbf{x} = \begin{bmatrix} x & y & t \end{bmatrix}^\top$, has a covariance function $k(\cdot, \cdot)$ that is a product of a spatial covariance function $k_s$, and a temporal covariance function $k_t$:

$$k\left(\begin{bmatrix}x\\y\\t\end{bmatrix},\begin{bmatrix}x'\\y'\\t'\end{bmatrix}\right)=k_s\left(\begin{bmatrix}x\\y\end{bmatrix},\begin{bmatrix}x'\\y'\end{bmatrix}\right)\cdot k_t\left(t,t'\right) \tag{2.9}$$

For example, a spatial phenomenon that exhibits smooth change over space, but less-smooth change over time, can be modelled by a Squared Exponential spatial covariance function, and a Matérn temporal covariance function. The input-scales of this process are modelled by the $\mathbf{P}$ matrices of the two covariance functions that compose it: for the spatial covariance function $\mathbf{P} = diag(l_s^2, l_s^2)$, with entries that determine the length-scale of the process, and for the temporal covariance function $\mathbf{P} = l_t^2$, containing a single entry encoding the time-scale of the process. For example, a spatial phenomena that varies slowly over space, but very quickly over time has a long length-scale $l_s^2$, but a short time-scale $l_t^2$.[4]

#### 2.3.2.1 Calculating GP regression with known Covariance function

When the covariance function of the GP is known, calculating predictions for arbitrary test inputs $\mathbf{y}_*$ using Equations 2.1 and 2.2 is fairly straightforward. However, explicitly inverting the covariance matrix of the training set $K(X, X)$ as suggested by these equations is very inefficient (especially if the training set is very large). Instead, we use a more efficient method that exploits the fact that the covariance matrix $\Sigma$ is symmetric and positive definite. This allows for the matrix to be decomposed into a product of a lower triangular matrix $L$ and its transpose: $\Sigma = LL^\top$, called a Cholesky decomposition. Given this decomposition, Algorithm 1 shows how the mean and variance of a set of test inputs can be efficiently computed.[5] However, when sequentially updating the GP regression with new training points (e.g. sensor observations), recalculating the Cholesky decomposition from scratch can become a computational bottleneck. In order to overcome this potential bottleneck, Osborne et al. (2008) present a number of numerical techniques for efficiently updating the Cholesky decomposition with newly acquired data points, as well as downdating it in order to discard old data points and to ensure that the size of the matrix $L$ stays within reasonable bounds. This not only saves memory and computational resources, but also ensures that predictions are based on the most recent (and relevant) part of the observation history.

[4] Note that the smoothness of the function along a dimension is distinct from the speed at which it varies along that dimension.

[5] For a $n \times n$ matrix $K(X, X)$, the Cholesky decomposition in line 3 takes $n^3/6$ operations, while solving the linear systems in steps 4 and 6 with triangular matrix $L$ takes $n^2/2$ operations. This is of significantly lower complexity than explicitly calculating $K(X, X)^{-1}$, which takes $n^3/3$ with Gauss-Jordan elimination.

**Algorithm 1** Prediction of values at coordinates $X_*$ given training data $X$, $\mathbf{y}$ and covariance function $k$ using the Gaussian process.

1: **Input:** matrix with training inputs $X$, training outputs $\mathbf{y}$, covariance function $k$, noise $\sigma_n^2$, $X_*$ test inputs. $K(X, X')$ denotes the matrix obtained by evaluating the covariance function $k$ for every pair of columns of $X$ and $X'$.
2: **Output:** $(\boldsymbol{\mu}, \Sigma)$ such that $\mathbf{y}* \sim \mathcal{N}(\boldsymbol{\mu}, \Sigma)$
3: $L \leftarrow cholesky(K(X, X) + \sigma_n^2 I)$
4: $\boldsymbol{\alpha} \leftarrow L^\top \setminus (L \setminus \mathbf{y})$
5: $\boldsymbol{\mu} \leftarrow K(X_*, X_*)^\top \boldsymbol{\alpha}$
6: $V \leftarrow L \setminus K(X, X)$
7: $\Sigma \leftarrow K(X_*, X_*) - V^\top V$

#### 2.3.2.2 Learning Covariance Functions from Data

In the previous section, we presented an algorithm for GP regression when the covariance function is completely known. This implies that the properties of the modelled phenomenon are completely known as well. In most realistic scenarios, however, the covariance function has to be learnt from data. In these cases, the use of GPs can be regarded as a three-level model selection problem. On the top level, the shape of the covariance function determines the high-level properties of the process. More specifically, it determines its smoothness (or differentiability), whether the process is periodic or isotropic. On the second level, the variables of the particular covariance function determine the extent to which these properties manifest themselves in the process. To illustrate this, consider again the noisy version of the Matèrn covariance function in Equation 2.8. In two spatial and one temporal dimensions, this covariance function has four free variables: the signal variance $\sigma_f$, the noise variance $\sigma_n$, the length-scale $l_s$ and the time-scale $l_t$. These variables are usually referred to as *hyperparameters*, since they determine the distribution of weights of an underlying parametric model (Rasmussen and Williams, 2006, Section 2.1). This parametric model constitutes the bottom level of the model selection problem, and is fitted to the data with Equations 2.1 and 2.2. In other words, we need only concern ourselves with the upper two levels of the selection problem, since the GP equations take care of the bottom level.

Now, in practise, the type of covariance function is usually chosen by an expert based on the most distinctive properties of the process. For example, as discussed earlier, we know that the Matèrn covariance function is an appropriate choice for modelling several environmental phenomena. However, the values of the hyperparameters are not known *a priori*, but have to be inferred from a set of observations. In practical terms, this involves the marginalisation of the hyperparameters $\phi$, a process captured in the following integral:

$$p\left(y_* | X_*, X, \mathbf{y}\right) = \frac{\int p\left(y_* | X_*, X, \mathbf{y}, \phi\right) p\left(\mathbf{y} | X, \phi\right) p(\phi) d\phi}{\int p\left(\mathbf{y} | X, \phi\right) p(\phi) d\phi} \tag{2.10}$$

In our case, the parameter space $\phi$ is the Cartesian product of the parameter spaces, i.e. $\sigma_f \times \sigma_n \times l_s \times l_t \subset \mathbb{R}^4$. Needless to say, in general, this space grows very large as the number of parameters increases; a phenomenon known as the *curse of dimensionality*. Furthermore, the non-trivial dependence between the likelihood of the parameters $p(\mathbf{y}|X,\phi)$ and the prediction $p(y_*|X_*,X,\mathbf{y},\phi)$ on $\phi$, causes this integral to become non-analytic. However, Osborne et al. (2008) show how sophisticated quadrature can be used to obtain a good approximation to Equation 2.10. In general, quadrature involves evaluating both $p(\mathbf{y}|X,\phi)$ and $p(y_*|X_*,X,\mathbf{y},\phi)$, for multiple samples of $\phi$, which is a computationally intensive operation.[6] Moreover, the size of the parameter space $\phi$ precludes exhaustive sampling from these functions. Therefore, *Bayesian Monte Carlo* (BMC) (Rasmussen and Ghahramani, 2003) has been proposed to reduce the number of samples. Unlike many frequentist approaches, BMC uses as much information about the sample functions as possible. So, instead of taking into account only the sample values, BMC also uses the sample locations and the smoothness of the integrand. As a result, BMC needs significantly fewer samples to produce an accurate approximation of $p(y_*|X_*,X,\mathbf{y})$. The result of applying BMC is a weighted sum of GPs:

$$p(y_*|X_*,X,\mathbf{y}) \approx \sum_{\phi} w_\phi p(y_*|X_*,X,\mathbf{y},\phi) \tag{2.11}$$

Note that $p(y_*|X_*,X,\mathbf{y},\phi)$ denotes a GP fitted to training data $\mathbf{y},X$ for hyperparameter sample $\phi$, and $w_\phi$ denotes the weight assigned to the GP with hyperparameters $\phi$. In other words, the prediction of $y$ is a weighted sum of predictions from GPs with the hyperparameters from the sample set. For a full treatment of BMC refer to Osborne et al. (2008) and Rasmussen and Ghahramani (2003). In this context, it suffices to say that by using algorithms for re-using the major part of the computations when new data points are collected, BMC is a suitable and efficient algorithm for learning the hyperparameters, while at the same time providing a principled means of performing regression and prediction.[7]

Thus far, in the GPs we have discussed, the covariance is a function of the difference of the input-vectors $|\mathbf{x}-\mathbf{x}'|$. These processes are referred to as *stationary*. Stationary processes are insensitive to translation. Consequently, these processes exhibit the same properties in different subsets of their input space. In many realistic scenarios, however, processes are sensitive to translation. For example, the temperature in a building will vary less rapidly in areas that are not exposed to sunlight than in areas that are. To deal with this challenge, several methods to model these non-stationary processes have been proposed, such as deforming the input-space (Guttorp and Sampson, 1994) and approximating

[6] Calculating $p(y_*|X_*,X,\mathbf{y},\phi)$ alone involves fitting a GP with parameters $\phi$ on the training set $\mathbf{y},X$ using Equations 2.1 and 2.2.

[7] Simpler alternatives for BMC exist for estimating hyperparameters $\phi$, such as Marginal Likelihood (ML) (Rasmussen and Williams, 2006, Section 5.4). However, our experiments showed that ML too often ends up in local maxima, resulting in very poor predictions.

the non-stationary GP with a mixture of one or more stationary processes (Nott and Dunsmuir, 2002). Unfortunately, non-stationary GPs involve more hyperparameters than a stationary GP, making them computationally more demanding. Nevertheless, the prevalence of non-stationary processes in realistic scenarios make them highly relevant in terms of our requirement of *accuracy*, since a non-stationary GP is more capable of modelling these phenomena more accurately. These challenges should therefore be addressed in future work. However, in the meantime, and in the remainder of this report, we will use stationary processes to model the environmental phenomena, in particular those that are governed by the Matèrn covariance function.

#### 2.3.2.3 Gaussian Processes in Related Work

Due to their versatility for modelling various kinds of processes, the use of GPs in sensor networks is reasonably widespread (e.g. Low et al. (2008)). Specifically, Guestrin et al. (2005) and Krause and Guestrin (2005) use the GP for modelling environmental phenomena with non-stationary covariance functions (using the techniques of (Nott and Dunsmuir, 2002)), with the aim of calculating highly informative placements for fixed wireless sensors. In order to do this, an initial deployment of a large number of sensors is required to collect a dataset of samples, from which the covariance function can be inferred. As an extension to this, Krause et al. (2006) also model the quality of the communication links between the sensors with a GP, to ensure a good trade-off between the informativeness of the sensor placement, and the expected communication cost between the sensors in the proposed deployment. However, both of these algorithms learn these models offline, and assume a dataset of samples is available prior to the deployment of the sensor nodes. In uncertain scenarios where time is of the essence, though, this assumption is too limiting. For instance, in our intended application scenarios, the mobile sensors will not be aware of the characteristics of their environment prior to the moment that they are introduced to it, let alone have access to a detailed set of samples.

So, in more time constrained and uncertain scenarios, such as those that are found in our intended application domains, the covariance function cannot be assumed to be known in advance. More specifically, for our solution to be *adaptive* (one of the requirements established in the introduction), sensors need to be able to learn the features of their surrounding environment *during deployment*. In GP terms, this implies that sensors need to be able to learn the covariance function online. To address this issue, Krause and Guestrin (2007) present an exploration-exploitation approach that learns the covariance function in an online fashion, while moving a mobile sensor towards more informative locations. In a similar vein, Osborne et al. (2008) perform online information processing on data streams from weather sensors using the techniques outlined in Section 2.3.2.2. Moreover, using a special kind of covariance function, they construct a GP with multiple correlated outputs, one for each sensor. With this GP, it is possible to predict missing

sensor data from a faulty sensor based on both readings obtained from the other (functional) sensors. This last property is especially attractive from the perspective of the requirement of *robustness*. As outlined in the introduction, robustness means that the operation of the mobile sensor network should not be interrupted when a single sensor fails. This technique allows the performance of the sensors to degrade gracefully, since the failure of sensors increases prediction error, but not catastrophically so.

### 2.3.3 Other Approaches

Besides the two regression methods discussed in the two previous sections, other techniques for information processing have been adopted in previous work. In what follows, these techniques are briefly covered.

As opposed to performing explicit regression, such as the techniques we covered earlier, Rahimi et al. (2004) subdivide the sensed area into strata of different sizes, such that the variance of the measurements taken within each stratum is below a certain threshold. When the variance within a stratum is found to be too large, it is divided into four substrata, and the same sampling method is recursively applied on each of them. As a result, the spatial resolution is adapted to the rate of change in different areas. The measurements obtained in this fashion constitute the reconstruction of the measured phenomenon. While this is an elegant and simple solution, it does not recover the spatial and temporal patterns that are present in the environment. As such, although it presents an accurate snapshot of the environment, it is not capable of extrapolating this snapshot in time or space, making it less suitable for the purpose of providing situational awareness.

In contrast, a common technique that does do justice to the underlying temporal dynamics of a phenomena is the Kalman filter. Thus far, we have considered information processing techniques for modelling phenomena when the underlying dynamics are unknown or highly uncertain. In many cases, however, these dynamics are known, and it is possible to predict how the current state of the system will transition into the next. For estimating the state of these types of dynamic processes from noisy observations, the Kalman filter has been found to be a suitable tool (Julier and Uhlmann, 1997; Durrant-Whyte et al., 1990; Deshpande et al., 2005b; Dash et al., 2005; Grocholsky et al., 2006). As opposed to the regression methods discussed in Sections 2.3.1 and 2.3.2, the Kalman filter is capable of explicitly taking into account the system dynamics. In target tracking, for example, it is known that the motion of a target is subject to the laws of Newtonian mechanics. Thus, based on a target's previous location, velocity and acceleration, it is possible to predict its current state. The Kalman filter combines this prediction with noisy range and bearing observations (for example from a radar) in order to refine the target's state estimation (a process that is called filtering). Consequently, when the dynamics of the system are known, the Kalman filter will be able to make more accurate

predictions than the GP, since it uses the known state transition model of the system, together with noisy observations obtained from the sensor.

Another attractive property of the Kalman filter is that it enables sensors to exchange and fuse beliefs about the state of their environment in a very communication efficient manner; sensors simply need to exchange their current state vectors, which are subsequently easily fused with other sensors' states to increase tracking accuracy. A drawback of the standard GP formulation, however, is that, unlike the Kalman filter, belief sharing involves exchange of observations.

In the environments that we aim to deploy our approach in, however, sensors are dealing with a possibly wide variety of environmental phenomena for which the temporal dynamics are either unknown, or highly complex. For this reason, we believe the Kalman filter is less suitable for modelling these phenomena. Instead, using the GP allows us to recover and approximate the intricate and complex dependencies between measurements in time and space with limited prior knowledge of the system dynamics.

In light of the above, a GP/Kalman filter hybrid, which combines the versatility of the GP and the communication efficiency of the Kalman filter would be desirable. Indeed, Reece and Roberts (2008) propose exactly such an approach, by showing the equivalence of the Kalman filter and GP. Further developments in this line of research could therefore be relevant a basis for a more flexible method of processing information.

## 2.4 Measuring Information Value

The information processing algorithms discussed in the previous section provide a high-level representation of the raw data, which enables sensors to reconstruct the dynamics of their environment. The quality of this reconstruction critically depends on the measurements that have been taken. To illustrate this with an example, consider a single sensor with a limited battery life. Suppose that this sensor starts sampling the temperature from its environment at its maximum rate. It will deplete its battery before the end of the day. Should we want to reconstruct the temperature during that day, the measurements collected by this sensor are less valuable than the samples collected by sensor that spaced its samples over the entire day. Put differently, the information that the sensor has collected is of little *value*.

Against this background, in the area of adaptive sampling, the concept of the *value* of information is of critical importance. This value is proportional to the extent to which it enables a sensor to provide a more accurate reconstruction of the monitored phenomenon. When the true value of the measurements that can be potentially collected in time or space is known, an adaptive sampling algorithm can decide where and when to sample. In making this choice, it has to make a trade-off between communication, movement and

sampling (e.g. excessive communication and movement will leave little battery power for sampling, and *vice versa*). However, the only way in which the true value of a sample can be known, is to collect it. Not surprisingly, this paradox presents a challenge. To address this challenge, we resort to an approximation of the value of a sample, based on the values of measurements taken in the past. The information processing techniques discussed earlier play an important role in this.

To this end, in this section, we will first give some background of how the challenge of measuring information value is addressed in previous work, and then show this affects the choices we made in our own work. In more detail, we demonstrate that all these different approaches are in some way based on a mathematical formalisation of the notion of expected "surprise" of a newly acquired measurement.[8] Equivalently, this can be thought of as the inverse of the confidence in the current model. The less confident an adaptive sampling algorithm is about its current model, the greater the potential surprise of a new sample, and the greater the value of that sample will be.

For example, Kho et al. (2009) use the distance of the confidence bands to the mean of the linear regression as a measure that is proportional to the value of information. As a result, when the environment changes at a higher rate, the linear model will start to deviate from the true dynamics of that environment. Consequently, new data points cannot be explained with the current model, and the sensor will be "surprised". The value of information will therefore be greater in dynamic situations than in those that are better characterised as static. In a similar vein, Padhy et al. (2007) model information value or surprise in terms of the Kullback-Leibler (KL) divergence between the prior and the posterior probability distribution of the linear model, i.e. the distribution over possible measurements before and after a new data point has been received, respectively. The larger this measure, the less capable the previous model was of explaining the new data point, and thus the more it has to be revised.

The same theme is found in approaches based on the GP. These generally exploit the fact that the covariance can be analytically evaluated (see Equation 2.2 for more details) to obtain a measure of information value.[9] For example, Osborne et al. (2008) use the predictive variance of a single future sample (see Equation 2.11) as an indicator of its information value. Not strictly related to sensor networks, but to observation selection in general, Ko et al. (1995) take the differential entropy of the observation as its information value (see appendix, Equation A.7). More formally, if we model a set of observations as a $D$ dimensional random vector $\mathbf{y}_*$ with a multivariate normal distribution (as is the case in the GP), the differential entropy $H$ is a function of its covariance matrix $\Sigma$ only:

[8] For the mathematical details of the information measures described in this Section, refer to Appendix A.2.

[9] If there exists uncertainty about the hyperparameters, the covariance is approximated using Equation 2.11.

$$H(\mathbf{y}_*) = \frac{1}{2} D \ln(2\pi e) + \frac{1}{2} \ln(|\Sigma|) \tag{2.12}$$

Informally, entropy is a measure of the peakedness of the probability distribution of a random variable: the more peaked the distribution, the less uncertainty is present in the variable, and the smaller the entropy will be, and the smaller the probability that we will be surprised by a new measurement. As shown by Guestrin et al. (2005), however, maximising entropy in a *greedy* way (i.e. by placing sensors one at a time at those locations with maximal entropy) leads to a fixed sensor deployment in which a large proportion of the sensors are placed along the border of the environment. Here, they are maximally uncertain about each others' measurements. Consequently, these sensors waste a large part of their sensor range. As an alternative, Guestrin et al. (2005) propose the mutual information (MI) criterion, which is a measure of uncertainty reduction: instead of measuring the entropy reduction at the location of the sensor alone, MI takes into account the reduction of the uncertainty in the rest of the environment:

$$MI(V \setminus L; L) = H(V \setminus L) - H(V \setminus L \mid L) \tag{2.13}$$

which they show leads to a more central placement of the sensors. Refer to Appendix A for more details on mutual information.

The implications to our research of the choices in previous work are twofold. First of all, in the context of situational awareness, "surprise" has a negative connotation, since it implies that there exist discrepancies between the decision maker's world picture and the real world. As a result, the decision maker risks basing his decisions on incomplete or wrong information. With this in mind, an adaptive sampling algorithm should be designed to minimise potential surprise as much as possible. This means that the way in which sensors value information should reflect this. Consequently, we adhere to the same rule found in the literature, stating that observations that are expected to maximally decrease uncertainty (and thereby potential surprise) should be valued the most.

Secondly—and in light of the previous point—it is attractive to adopt the Mutual Information measure used by Guestrin et al. However, our experiments with greedy algorithms for mobile sensors showed that the calculation of MI is a computationally costlier process than calculating the entropy (see Section 4.3). The reason for this is that it requires the calculation of the entropy reduction in the entire environment for every possible movement, instead of only the entropy reduction obtained at the new position of the sensor. More importantly, however, these experiments showed no significant improvements in performance when using MI instead of entropy. We are not entirely certain why our findings differ from those made by Guestrin et al., who showed that the entropy criterion leads to suboptimal use of the sensors' range. However, we hypothesise that this results from the fact that we not only consider the spatial dimension of the observation selection

problem, but also the temporal dimension (see Section 2.2.1). As a result, the mobile sensors are positioned in a three-dimensional continuum, in which they only control their movements along the two spatial dimensions. These two considerations (i.e. less computation with equivalent accuracy) have led to our adoption of the entropy criterion in favour of the MI criterion.

Having reviewed several information processing methods used in previous work, and the value of information derived from these methods, we can now turn to the third and final aspect of an adaptive sampling algorithm: maximising the sample value from a collection of sensors.

## 2.5 Maximising Information Value

In the previous two sections we looked at information processing and valuing information. In this section, we address the third and final challenge of an adaptive sampling algorithm: maximising sample value. Here, the information processing and information valuing methods are exploited to determine where and when the sensors should sample. Reviewing the state of the art, we can distinguish between two different approaches: offline adaptive sampling and online adaptive sampling. The former algorithms plan in advance when and where sensors should take measurements. In contrast, the latter do not perform explicit planning, but instead base their next decision on the data that has been gathered so far. Clearly, the former is more suitable when prior knowledge of the environment is available, such as physical layout, and, for example, the spatio-temporal correlations between the measurements that are taken by the different sensors, allowing sensors to pre-plan their paths. Online adaptive sampling algorithms are more suitable when there exists uncertainty about the environment, or where unexpected events are likely to influence the operation of the sensor network, since they can adapt to the prevailing circumstances.

In this section, we first discuss both approaches in more detail from the point of view of a single sensor. Then, we will specifically highlight the challenge of coordinating the actions of multiple sensors, which is a key issue in our work.

### 2.5.1 Offline Sensing Algorithms

Offline sensing algorithms based on GPs are commonly applied in scenarios when the covariance function and the value of the hyperparameters are known before deployment of the sensors. To this end, several authors propose offline algorithms for calculating informative sensor placements (see Section 2.3.2.3). More specifically, these algorithms greedily maximise a placement criterion, such as mutual information. The reason for doing this in a greedy fashion is because the problem of calculating a sensor placement

that maximises mutual information (or entropy, for that matter) is NP-hard (Ko et al., 1995). So instead of solving this problem optimally, an approximate greedy algorithm is used to deploy sensors one at a time, whereby the location $v$ of each sensor is chosen so as to maximise MI in the rest of the environment $V = \{v_1 \dots v_n\}$ given the set of locations of the previously deployed sensors $S \subset V$:

$$v = \arg\max_{v \in V \setminus S} MI(V \setminus (S \cup \{v\}; S \cup \{v\}) \tag{2.14}$$

In this context, Guestrin et al. (2005) show that this greedy algorithm leads to a lower bound on the solution quality of $(1 - 1/e)$ of the optimal solution, due to the submodularity of MI.[10] Moreover, whereas finding the optimal solution is an NP-hard problem, this greedy algorithm has polynomial complexity. In addition to maximising information gain, Krause et al. (2006) simultaneously attempts to minimise communication cost between the deployed nodes. In order to do this, their algorithm exploits the *locality* property of the environment. This means that the correlation between measurements taken at two distant locations is small enough to assume they are independent.[11] Using this property, the problem is first decomposed to calculate sensor placements for different subspaces of the environment. Next, these individual placements are connected whilst ensuring the communication costs do not exceed a given threshold (in the budgeted case), or the sensing quality exceeds a given threshold (the quota problem).

Singh et al. (2007) and Meliou et al. (2007) both extend the work of Krause et al. by exploiting the same properties of submodularity and locality for pre-planning informative paths for mobile robots. A similar decomposition strategy is used as in the sensor placement algorithm, whereby the environment is divided into grid cells that can be considered independent because of the locality assumption. Their algorithms then perform a search over paths by connecting grid cells that contain as much information as possible, and return the robot to its original location. Relatedly, the approach by Zhang and Sukhatme (2007) also uses mobile robots for sensing, and performs a breadth-first search through the space of all feasible paths. In their case, a path is feasible if the boat has sufficient energy to follow it, and does not not return to previously visited locations. The path that maximises the cumulative information gain obtained at the locations visited along that path is returned.

As already hinted at in Section 2.3.2.3, these offline algorithms for both fixed and mobile sensors suffer from the same drawback: they assume that the hyperparameters of the

[10] A set function $f$ is called submodular, if $f(B \cup Y) - f(Y) \leq f(A \cup Y) - f(Y)$ for $A \subset B$, i.e. adding an element to a smaller set increases the value of $f$ more than adding the same element to a larger set. Applied to the problem of sensor deployment, this means that adding a sensor to a small deployment of sensors increases the quality of that deployment more than adding it to a large deployment of sensors. Nemhauser and Wolsey (1978) determine several lower bounds on the solution found by greedy maximisation of submodular functions, on which the work by Guestrin et al. (2005) is based.

[11] For example, with the Squared Exponential covariance function in Section 2.3.2, the covariance between the outputs drops exponentially fast as the distance between their input-vectors increases.

process they are modelling are known in advance. While this is not necessarily a problem for fixed wireless sensor networks, we believe that this is a somewhat contradictory assumption when taking one-off measurements with mobile robots. After all, from a practical point of view, when the covariance structure of the environment is known in advance, it is unclear why these measurements should be taken in the first place.

Furthermore, from the point of view of *adaptiveness* and *autonomy*, pre-planning paths through the environment does not allow sensors to adjust their paths based on unexpected events. Moreover, these algorithms compute these paths in a centralised way, after which the solution is distributed over the mobile robots, increasing their vulnerability to failure of the central controller, thus which in turn decreases their *robustness*. In light of the requirements defined in the introduction, these properties make them less suitable for our purposes, especially given the robustness requirement; should one sensor fail, the algorithm is unable to adapt, resulting in unforeseen degradation of performance. Given this, we will examine online approaches in the next section, that are better able to maximise information gain in the presence of more uncertain and dynamic phenomena. More specifically, we will see how the greedy offline algorithms discussed in this section can be modified to better suit these circumstances, and how they form the basis of our greedy algorithm presented in Chapter 4, and the negotiated coordination algorithm presented in Chapter 5.

### 2.5.2 Online Sensing Algorithms

In contrast to offline algorithms, their online counterparts do not plan (sensing) actions before the sensor network has been deployed. Instead, they adapt their sampling to the prevalent conditions within the environment. Sampling is constantly revised to reflect the lessons learnt from the sampling history. For example, the expected information value from the next sample moment is determined from the information value associated with actual samples taken in the past.

The adaptive sampling algorithm by Padhy et al. (2007) is a good example of this idea. In their algorithm, each sensor extrapolates the value of the information it has obtained so far to estimate when it is worthwhile to take a new sample. Furthermore, the decision about whether to send a particular datum to the WSN's base station is based on the opportunity cost of communicating. Here, opportunity cost is expressed as the amount of information value that could have been obtained if the sensor decides not to communicate, i.e., if the sensor spends the energy required for communication on collecting additional samples. This applies not only to sensors where the data has originated, but also to sensors that act as a relay station for multi-hop communication with the base station. When the opportunity cost of sending a datum is low compared to the value associated with that datum, a decision is made in favour of communicating. The resulting behaviour

of this algorithm can be characterised as emergent: the simple local rules give rise to more complex and desired behaviour.

Emergent behaviour is also found in the work of Kerr and Spears (2005). Their work is aimed at covering an environment with a large number of mobile robots (called a *swarm*), without an explicit measure of information. These robots have very limited global knowledge, and are only aware of their neighbours' locations. This is achieved by localising them through emitting RF and acoustic pulses. Inspired by gas and fluid models, the robots can determine their speeds and headings after a robot-robot or robot-wall collision, thereby effectively emulating the way gas particles disperse in an environment. It is shown that from the properties of gasses, the mobile robots proceed to scatter and position themselves evenly throughout the environment. Consequently, this method is especially attractive when a large number of robots is available. Moreover, this approach is strong in terms of robustness, because a single robot failure, will not cause the performance of the remaining robots to suffer. However, when only a few robots are present in the environment, its performance might degrade rapidly, since the collisions with other robots will become less likely. The resulting coverage will be less-than-optimal, and because the sensors do have no measure of information that is related to the rate at which their environment is changing, the world model that results from their movements will not be able provide good situational awareness.

As mentioned in Section 2.5.1, the offline algorithms for path-planning suffer from the lack of *adaptiveness*, since plans are precomputed and thus unable to react to unforeseen circumstances. To address this shortcoming, Low et al. (2008) combine the use of Gaussian processes to represent the environment, with the use of Markov decision processes to compute non-myopic paths for multiple mobile sensors in an *online* fashion. Whilst such a non-myopic approach avoids the problem of local minima (and thus computes high quality solutions), it incurs significant computational cost (it is only empirically evaluated for systems containing just two sensors), and is a centralised solution, thereby failing to provide a *scalable* and *robust* coordination algorithm.

Instead, our work is inspired by two related branches of research, the first of which is artificial potential fields. As described in the work of Kerr and Spears (2005), artificial potential fields do not require planning, and can be considered reactive. However, unlike their work, a potential field does have a notion of information value. Originally proposed for motion planning of robots (e.g. Dunias (1996)), these potential fields can be used to attract mobile sensors to informative areas, while repelling them from areas that are highly certain. For example, Pereira et al. (2004) uses this technique for controlling a mobile robot that collects data from fixed sensors. The robot is attracted to the fixed sensors that collect samples at a high rate, and consequently have collected a large amount of data. Once the data has been downloaded, the potential field disappears, and the robots sets its heading toward the sensor that emits the next strongest potential field. The techniques used in the work of Grocholsky (2002) and Grocholsky et al. (2005,

2006) also bear some resemblance to potential fields. Specifically, the sensor platforms continuously update the amount of information that is associated with observing the target from different locations. When locations of equal information content are connected with contour lines, it becomes possible to determine the direction in which the amount of information increases the most. This is done each time a new localisation of the target has been made. Grocholsky refers to this technique as *information surfing*.

The second branch of research that inspired the work in this report is based on GPs. In more detail, Krause and Guestrin (2007), develop a myopic control algorithm with strong theoretical bounds by extending their earlier work on fixed sensor placement (Guestrin et al., 2005; Krause and Guestrin, 2005; Krause et al., 2006). More specifically, they develop a strategy in which a single mobile sensor is capable of learning the hyperparameters online. Recall from the previous section that their sensor placement algorithms assume that the hyperparameters of the GP modelling the environmental phenomena are known beforehand. In many scenarios, this assumption does not hold. In these cases, the hyperparameters need to be learnt online, while at the same time repositioning the sensor to increase the information gain. Moreover, as discussed in Section 2.3.2.1, the (co)variance of a GP is only independent of the actual observations if the covariance function is known. However, when hyperparameters need to be learnt from actual data, this independency no longer holds, because the variance depends on the values of hyperparameters through the covariance function. These considerations have led to a *greedy, online* policy that is the focus of Chapter 4.

### 2.5.3 Coordination of Multiple Sensors

So far, we have discussed information maximisation algorithms for teams of sensors without explicitly focussing on coordination. In this section, we will address the challenge of coordinating multiple sensors, in order to ensure that their collective action results in a significant collective information gain. The importance hereof is mostly evident in online algorithms that do not calculate sensing schedules or motion paths at design time. Consequently, the sensors need to be able to collectively determine when and where to sample at run time, in order to maximise the use of the resources at their disposal. Now, in many cases, a centralised view of the environment is not available, or the costs associated with obtaining such a view are prohibitive (see the discussion of the *autonomy* property in Section 1.1). In these situations, there is a necessity for *decentralised* algorithms. These algorithms are characterised by the fact that each sensor makes individual decisions from information that is available locally, or through message passing with direct neighbours. As a result, the team is robust against failures (since no central point of failure exists), each sensor controls its own actions (and is autonomous), and the amount of computation that an individual sensor needs to perform scales with the number of its neighbours—not with the size of the team—thus ensuring the scalability

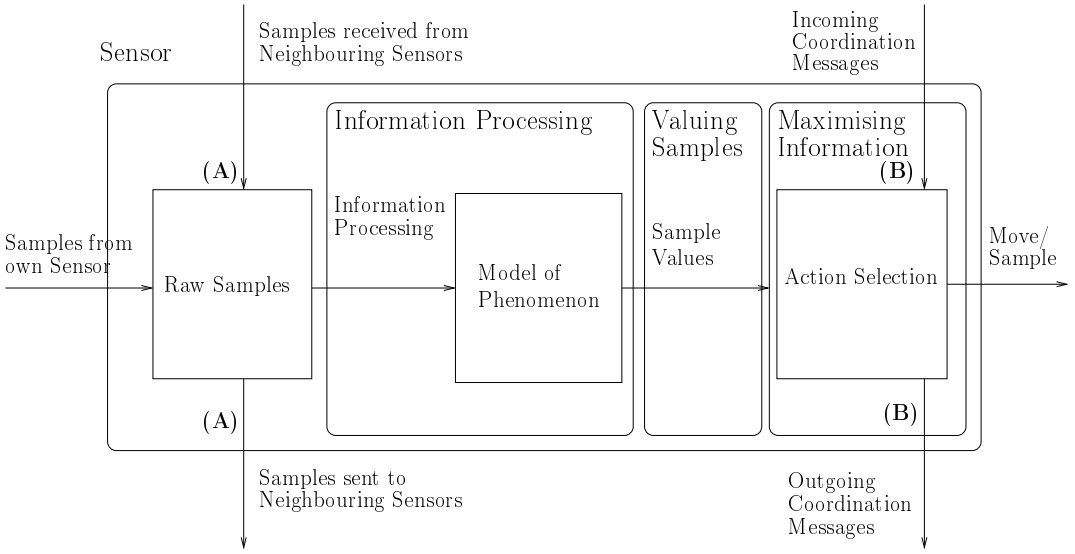



FIGURE 2.4: A general architecture for a sensor node.

of the solution. Clearly, all these properties are desirable in terms of the requirements stated in the introduction.

Now, in the literature, three levels of coordination are distinguished (Grocholsky, 2002). Ordered by a decreasing amount of shared belief, these are:

1. Cooperation or Negotiated Coordination — sensors share a common world view, and negotiate about their actions.

2. Un-negotiated Coordination — sensors have a shared world view, but individually decide how to optimise their utility, without negotiating with their neighbours.

3. Passive Coordination — sensors do not communicate, but coordinate through passive coordination mechanisms, for example by observing each other's position or behaviour.

Returning to the architecture of the general sensor node introduced at the start of this chapter, which we repeat in Figure 2.4, we can identify how the three levels of coordination impact on it. Starting with the lowest level of coordination, Passive Coordination neither requires belief sharing, nor the exchange of negotiation messages. Consequently, arrows **A** and **B** are not present in the architecture of a sensor operating in this mode. Moving up one coordination level, Un-negotiated Coordination shares beliefs, but does not explicitly negotiate about actions. So, while arrows **A** are present for Un-negotiated Coordination, arrows **B** are not. Finally, in Negotiated Coordination mode, sensors both share beliefs, and negotiate about actions. As a result, both arrows **A** and **B** are present.

In the negotiated cooperation and the un-negotiated coordination modes, sensors share a common world view through the exchange of messages. These messages are related to

the observations that they have made. However, these messages do not need to contain only raw samples obtained from the environment, but can also contain a summary or aggregation of the samples made so far. In particular, it is desirable to make the message size independent of the amount of samples its contents are based on. For example, the use of the information variant of the Kalman filter lends itself to efficient communication, since only the sensor's current belief needs to be communicated with others (Grocholsky et al., 2006; Reece and Roberts, 2005). To the best of our knowledge, there does not exist any work that achieves a similar efficiency for the distributed use of the GP.[12] For instance, Krause and Guestrin (2007) do not consider the multi-sensor case, and, consequently, do not address how their methods could be distributed over multiple sensors.

The crucial difference between cooperation in negotiated or un-negotiated mode is the fact that each sensor unilaterally decides their best action in the latter mode, without informing the other sensors of its intentions. As a result, the sensors will only be aware of the impact of the collective decision after these have been made. In negotiated cooperation, however, sensors will only execute their preferred action after a negotiation phase with their neighbours. This negotiation is aimed at maximising the collective utility of the sensors, which is commonly expressed in terms of the trade-off between information gain and the cost associated with the action. For instance Ahmadi and Stone (2006) use negotiation to assign partitions of their environment to mobile sensors in the context of event detection. Once a sensor has swept the partition it was allocated to, it negotiates with sensors in adjacent partitions to see whether a repartitioning of the environment results in a better distribution of tasks, and thus in an increase of the probability of detecting an event. Whilst being an attractive approach, it assumes an initial partitioning of the environment is present, and it is unclear how the algorithm would adjust to a change in the layout of the environment (for example, caused by the sudden obstruction of a doorway). Additional work on coordination between multiple distributed entities is abundant, and includes several game theoretic formulations of the problem (Arslan et al., 2007), or the application of algorithms based on the Generalised Distributive Law (Aji and McEliece, 2000). Because of the importance of the latter in the context of our work, the next section will provide a more in-depth discussion of these algorithms.

#### 2.5.3.1 The Max-Sum Algorithm

In the negotiated cooperation mode mentioned above, a team of sensors collaboratively attempt to maximise team performance through a process that involves the exchange of messages, before a decision is reached. Depending on how team utility is derived from the actions of individual sensors, various existing algorithms can be selected to maximise team utility in a distributed fashion.

[12] Although the methods proposed by Reece and Roberts (2008) seem very promising. As mentioned before, an investigation into the applicability of these methods is part of future work.

In particular, it is often the case that team utility can be *factorised* into a sum of individual sensors' utilities. This is commonly referred to as *social welfare* within the multi-agent systems literature.[13] Within this setting, we wish to find the value of each sensor's control parameter, $\mathbf{p}^*$, such that the sum of the individual sensors' utilities is maximised:

$$\mathbf{p}^* = \arg\max_{\mathbf{p}} \sum_{i=1}^{M} U_i(\mathbf{x}_i) \tag{2.15}$$

Furthermore, in order to enforce a decentralised solution, we assume that a sensor only has knowledge of, and can directly communicate with the few neighbouring agents that influence its utility directly. As a result, the complexity of the computation that an agent has to perform depends on the number of neighbours it has, and not on the total number of sensors in the team. Consequently, we can achieve solutions that scale well. Clearly, these properties are highly desirable in light of the requirements of scalability, robustness and autonomy.

With these considerations in mind, we can treat Equation 2.15 as a Distributed Constraint Optimisation Problem (DCOP) (Modi et al., 2003), in which "multiple cooperative agents control one or more variables and work together to optimise a set of constraints that exists upon these variables"[14]. In recent years, this type of problem has been studied extensively, which has led to a wide range of algorithms that can be readily applied to solve them. Such algorithms can be broadly divided in two main classes: complete algorithms that generate optimal solutions such as ADOPT (Modi et al., 2005), OptAPO (Mailler and Lesser, 2004), and DPOP (Petcu and Faltings, 2005); and approximate algorithms such as the Distributed Stochastic Algorithm (DSA) (Fitzpatrick and Meertens, 2003) or Maximum Gain Message (Maheswaran et al., 2005).

Now, while complete algorithms provide guarantees on the solution quality, they also exhibit an exponentially increasing coordination overhead (either through the size and/or number of messages exchanged, or in the computation required by each sensor) as the number of sensors in the network increases. Conversely, approximate algorithms require very little local computation or communication, but often converge to poor quality solutions because the sensors within the network do not propagate information across the whole network. Rather, local information is only used by neighbouring sensors. For example, in DSA each sensor communicates its preferred action (e.g., the one that will maximise its own utility) based on the current preferred actions of its neighbours only.

However, there exists a class of algorithms usually referred to under the framework of the Generalised Distributive Law (Aji and McEliece, 2000), that constitute a compromise

[13] The same problem is also referred to as the optimal control problem in control theory (Paskin et al., 2005).

[14] `http://teamcore.usc.edu/dcop/`

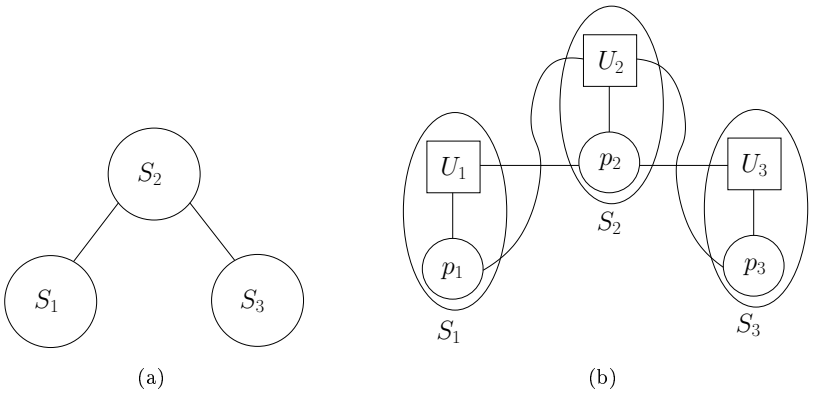



FIGURE 2.5: Diagram showing (a) the interactions of sensors, $S_1$, $S_2$ and $S_3$, (b) factor graph representing the sensors' utility.

between the extremes represented by these two classes, and can be used to obtain good approximate solutions. These algorithms have been widely used in the field of information theory and probabilistic inference to decompose complex computations on single processors (MacKay, 2003), and more recently both complete and approximate algorithms from this framework have been applied to the coordination of networked sensing devices within the domain of discrete control parameters (Paskin et al., 2005; Farinelli et al., 2008). In particular, one of the approximate algorithms, based upon the max-sum algorithm, has been shown to generate solutions closer to the optimum than previous approximate stochastic DCOP algorithms (Farinelli et al., 2008). It does so with an acceptable computation and communication overhead when benchmarked against representative complete algorithms (specifically DPOP), and it has been shown to be robust to message loss. Due to the fact that this algorithm exhibits these properties, it has been deployed and validated on low-power embedded devices.

In more detail, the max-sum algorithm operates on a *factor graph* that represents optimisation problem described in Equation 2.15. A factor graph is an undirected bipartite graph in which vertices represent variables $p_i$ and utility functions $U_j(\mathbf{p}_j)$. In such factor graphs, an edge exists between a variable $p_i$ and a function $U_j$ iff $p_i \in \mathbf{p}_j$, (i.e., $p_i$ is a parameter of $U_j$). For example, Figure 2.5(a) shows three interacting sensors, $S_1$, $S_2$ and $S_3$, and the resulting factor graph consisting of variable and function nodes representing each sensor's control parameter and utility is shown in Figure 2.5(b). In order to have a truly decentralised computation, the function that represents a sensor's utility, as well as the variable that represents the sensor's control variable are assigned to the physical computational unit associated with that sensor.

Now, in a team of $M$ sensors, sensor $S_i$ first computes:

$$\tilde{U}_i(p_i) = \max_{\mathbf{p} \setminus p_i} \sum_{i=1}^{M} U_i(\mathbf{p}_i) \tag{2.16}$$

in a distributed way (i.e. based on local information and communication with direct neighbours), after which the optimal decision $p_i^*$ is obtained as follows:

$$p_i^* = \arg\max_{p_i} \tilde{U}_i(p_i) \tag{2.17}$$

Function $\tilde{U}_i(p_i)$ is computed by message passing between the functions $U_i$ and the variables in $\mathbf{p}_i$ as follows:[15]

- **From variable to function:**

$$Q_{p_n \to U_m}(p_n) = \sum_{\substack{U_{m'} \in adj(p_n) \\ U_{m'} \neq U_m}} R_{U_{m'} \to p_n}(p_n) \tag{2.18}$$

- **From function to variable:**

$$R_{U_m \to p_n}(p_n) = \max_{\mathbf{p_m} \setminus p_n} \left[ U_m(\mathbf{p}_m) + \sum_{\substack{p_{n'} \in adj(U_m) \\ p_{n'} \neq p_n}} Q_{p_{n'} \to U_m}(p_{n'}) \right] \tag{2.19}$$

The messages flowing into and out of the variable nodes within the factor graph are functions of a single variable that represent the total utility of the factor graph for each possible value of that variable. To interpret the meaning of these messages, note that variables summarise the utility that is obtained in adjacent parts of the factor graph using the $\Sigma$ operator. Functions have to perform more intricate computation; the expression that is maximised between brackets is the sum of a local component (the utility function) and a component that pertains to the utility obtained in adjacent parts of the factor graph (i.e. the messages from variables). The function node then finds the maximum possible value after fixing a single variable (i.e. $p_n$).

At any time during the propagation of these messages, sensor $i$ is able to determine which action it should take such that the sum over all the sensors' utilities is maximised. This is done by locally calculating the function, $\tilde{U}_i(p_i)$ from Equation 2.16, from the messages flowing into sensor $i$'s variable node:

$$\tilde{U}_i(p_i) = \sum_{U_m \in adj(p_n)} R_{U_m \to p_n}(p_n) \tag{2.20}$$

and hence finding:

$$p_i^* = \arg\max_{p_i} \tilde{U}_i(p_i) \tag{2.21}$$

[15] In what follows, we use $adj(U_m)$ to denote adjacent vertices of function $U_m$ in the factor graph, i.e., the set of variables in the domain $\mathbf{p}_m$ of $U_m$. Similarly, $adj(p_n)$ denotes the set of functions in which $p_n$ occurs in the domain.

Now, when the factor graph is cycle free, the algorithm is guaranteed to converge to the global optimal solution such that it finds the combination of actions that maximises the sum of the sensors' utilities (MacKay, 2003). When applied to cyclic graphs, there is no guarantee of convergence, but extensive empirical evidence demonstrates that this family of algorithms generate good approximate solutions (Kschischang et al., 2001; Frey and Dueck, 2007).

In more detail, when max-sum is applied to a cyclic factor graph, it becomes necessary to normalise the messages in Equation 2.18 to prevent values from growing arbitrarily large. More specifically, a constant $\alpha_{nm}$ is added that is chosen such that $\sum_{p_n} Q_{n\to m}(p_n) = 0$. Whilst normalisation breaks equality in Equation 2.16 and 2.20, it does not influence the preference ordering encoded by $\tilde{U}_i(p_i)$, and the solution obtained from Equation 2.21 remains unchanged.

The messages described above may be randomly initialised, and then updated whenever a sensor receives an updated message from a neighbouring sensor; there is no need for a strict ordering or synchronisation of the messages. In addition, the calculation of the marginal function shown in Equation 2.20 can be performed at any time (using the most recent messages received), and thus, sensors have a continuously updated estimate of their optimum action.

The final result of the algorithm depends on the structure of the sensors' utility functions, and, in general, three behaviours can be observed:

1. The preferred actions of all sensors converge to fixed actions that represent either the optimal solution, or a solution close to the optimal, and the messages also converge (i.e. the updated message is equal to the previous message sent on that edge), and thus, the propagation of messages ceases.
2. The sensors' preferred actions converge as above, but the messages continue to change slightly at each update, and thus continue to be propagated around the network.
3. Neither the sensors' preferred actions, nor the messages converge and both display cyclic behaviour.

Thus, depending on the problem being addressed, and the convergence properties observed, the algorithm may be used with different termination rules:

1. Continue to propagate messages until they converge, either changing the action of the sensors continuously to match the optimum indicated, or only after convergence has occurred.

2. Propagate messages for a fixed number of iterations per sensor (again either changing the action of the sensor continuously or only at termination).

The first termination rule favours the quality of the solution. When the algorithm converges, it does not converge to a simple local maximum, but to a neighbourhood maximum that is guaranteed to be greater than all other maxima within a particular large region of the search space (Weiss and Freeman, 2001). Depending on the structure of the factor graph, this neighbourhood can be exponentially large. However, only limited guarantees for convergence of the max-sum algorithm exist, and for general factor graphs the algorithm might not converge. For practical applications, therefore, the second termination rule is often preferred. In fact, empirical evidence shows that the max-sum algorithm reaches good approximate solutions in few iterations. In addition, in dynamic scenarios where the utilities of the sensors or the interactions between them change over time, the max-sum algorithm can run indefinitely without any termination rule; each sensor can decide at every cycle which action to choose based on Equation 2.20, and operates on a continuously changing coordination problem.

In sum, due to its robustness to message loss, the fact that it is fully decentralised, and that it has an acceptable computation and communication overhead, the max-sum algorithm is an attractive option to serve as a basis for a coordination algorithm to control mobile sensors. In Chapter 5, we will therefore show how max-sum can be applied to the sensor monitoring problem, resulting in a decentralised coordination algorithm that has many of the required properties stated in Chapter 1.

## 2.6 Summary

Let us recapitulate what we have discussed in this chapter. We have reviewed the literature along two dimensions. In terms of the application dimension, we have seen that the practical use of WSNs spans a large set of applications. Along the technical dimension, we reviewed previous work on adaptive sampling in terms of the three central challenges that need to be addressed: information processing, information valuing, maximising information value.

Although these challenges are addressed in previous work in different ways, we have shown that several patterns and similarities between different approaches can be distinguished. More importantly, this review has allowed us to identify the techniques that can be adopted in our own work.

In particular, we will use the same break-down into the three constituent components of an adaptive sampling algorithm—information processing, information valuing, and information maximisation—to give a brief overview of the choices made in this chapter. Firstly, for information processing, we have seen that the GP is a very versatile

and powerful technique for accurately modelling the spatial and temporal dynamics of various environmental phenomena. Specifically, from the perspective of our requirement of accuracy, insight on these dynamics for achieving situational awareness is important, and the GP allows us to recover them in a straightforward way. Moreover, the techniques proposed by Osborne et al. (2008) make it possible to do this in an online fashion by updating the existing GP models with newly obtained observations in an efficient way. In so doing, the GP satisfies the requirement of adaptiveness as laid down in the introduction. However, the versatility of the GP comes at a higher computational cost than, say, linear regression. Moreover, future work will need to address the challenge of distributing the use of the GP over multiple sensors. Nevertheless, we believe the GP remains an attractive technique, since it does not put restrictions on the class of phenomena that can be modelled, which is an especially desirable property at this stage of our work.

Secondly, we showed that the value of information is connected to the notion of surprise. Moreover, since surprise in situational awareness is something that needs to be avoided, the value of information should be chosen to favour observations that minimise surprise, and maximise the accuracy of the world picture of the sensors. We believe that entropy is a fitting choice at this stage, since it is cheap to compute. Moreover, initial experiments showed favourable performance of the entropy criterion in comparison with the MI criterion, which has been extensively evaluated in the literature.

Thirdly, we have expressed our preference for myopic information maximisation algorithms, since they are highly adaptive, which makes them very suitable for situational awareness in dynamic environments. Moreover, they are governed by simple rules, that instruct sensors to choose their actions myopically to maximise entropy. This makes a simple decentralised implementation on a multi-sensor platform possible, while ensuring that the individual sensors are autonomous, and that the overall performance of the collective is robust against failure of single sensors. Put differently, these are attractive properties in terms of our requirements. Given this, in Chapter 4 we discuss a *un-negotiated* (see Section 2.5.3) algorithm based on these ideas.

Finally, we discussed various coordination mechanisms. In particular, we focussed on the max-sum algorithm for decentralised coordination and argued that it exhibits desirable properties such as explicit coordination, robustness, and autonomy. In Chapter 5 we extend the *un-negotiated coordination* algorithm from Chapter 4 by applying the max-sum algorithm to obtain a *negotiated coordination* algorithm, and show that this leads to improved performance. In Chapter 6, we extend this algorithm to the case where the sensors are characterised by continuous control parameters, by generalising the max-sum algorithm from discrete to continuous action spaces.

# Chapter 3

# The Mobile Sensor Monitoring Problem

.

In this section we present a formalisation of the environmental monitoring problem for multiple mobile sensors, and the assumptions on which this formalisation is based. This formalisation is inspired by Meliou et al. (2007), and has been extended for multiple sensors with limited local knowledge. In Chapters 4, 5 and 6 we develop several algorithms that enable a team of sensors to solve this problem in a decentralised fashion.

Intuitively, an environment is defined by its physical layout, and by the phenomena (temperature, humidity, etc.) that exist within. More formally, we can denote an environment and the mobile sensors within it by a tuple $\mathcal{E} = (\mathcal{S}, \mathcal{G}, \mathcal{P}, T)$, where:

- $\mathcal{S} = \{S_i | i = 1 \dots M\}$ is the set of $M$ mobile sensors;
- $\mathcal{G} = (V, E)$ encodes the layout of the physical environment, where $E$ denotes the possible movements between locations $V$. To each location $v \in V$ we assign two spatial coordinates: $(x_v, y_v)$. Thus, with some abuse of notation, we can denote the distance between two locations $v$, $v'$ as $|v - v'|$;
- $\mathcal{P}$ is a spatial phenomena that is monitored by the sensors in $\mathcal{S}$. Here, we model phenomenon $\mathcal{P}$ as a scalar field defined on one temporal and two spatial dimensions: $\mathcal{P} : \mathcal{V} \times T \rightarrow \mathbb{R}$.
- $T = \{t_1, t_2, \dots\}$ models time as a sequence of discrete timesteps of unknown length.[1]

[1] In uncertain and dynamic scenarios, the mission time is often not known beforehand.

Furthermore, we denote the sensors' locations at time $t \in T$ by the $M$-tuple $\mathcal{L}_t = \left(l_t^1, \ldots, l_t^M\right)$, where $l_t^i \in \mathcal{V}$. At every timestep $t \in T$, the sensors take measurements $O_t = \left(o_t^1, \ldots, o_t^M\right)$ at locations $L_t$ by sampling from $\mathcal{P}$: $o_t^i \leftarrow \mathcal{P}(l_t^i, t)$, and move to a new location that is adjacent to the current location in $V$: $l_{t+1}^i \in adj_{\mathcal{G}}(l_t^i)$. To illustrate this formal model with an example, Figure 3.1 shows the first few timesteps of a team of four sensors moving an example environment.

Given this model, the sensors' challenge is to monitor $\mathcal{P}$ at all locations $\mathcal{V}$ at time $t$. Since the number of sensors $M$ is generally much smaller than $|V|$ (the number of locations that are monitored), the sensors need to not only take measurements at locations $\mathcal{L}_t$, but also *predict* the value of $\mathcal{P}$ at time $t$ for every location $V$, based on observations made earlier. In order to do this, we associate to the measurement at location $v \in V$ at time $t$ a continuous random variable $\mathcal{X}_{v,t}$, and use a statistical model to predict values at locations $V$. As we discussed in the previous Chapter, we will model the phenomenon $\mathcal{P}$ with a GP (see Section 2.3.2), that encodes both its spatial and temporal correlations.

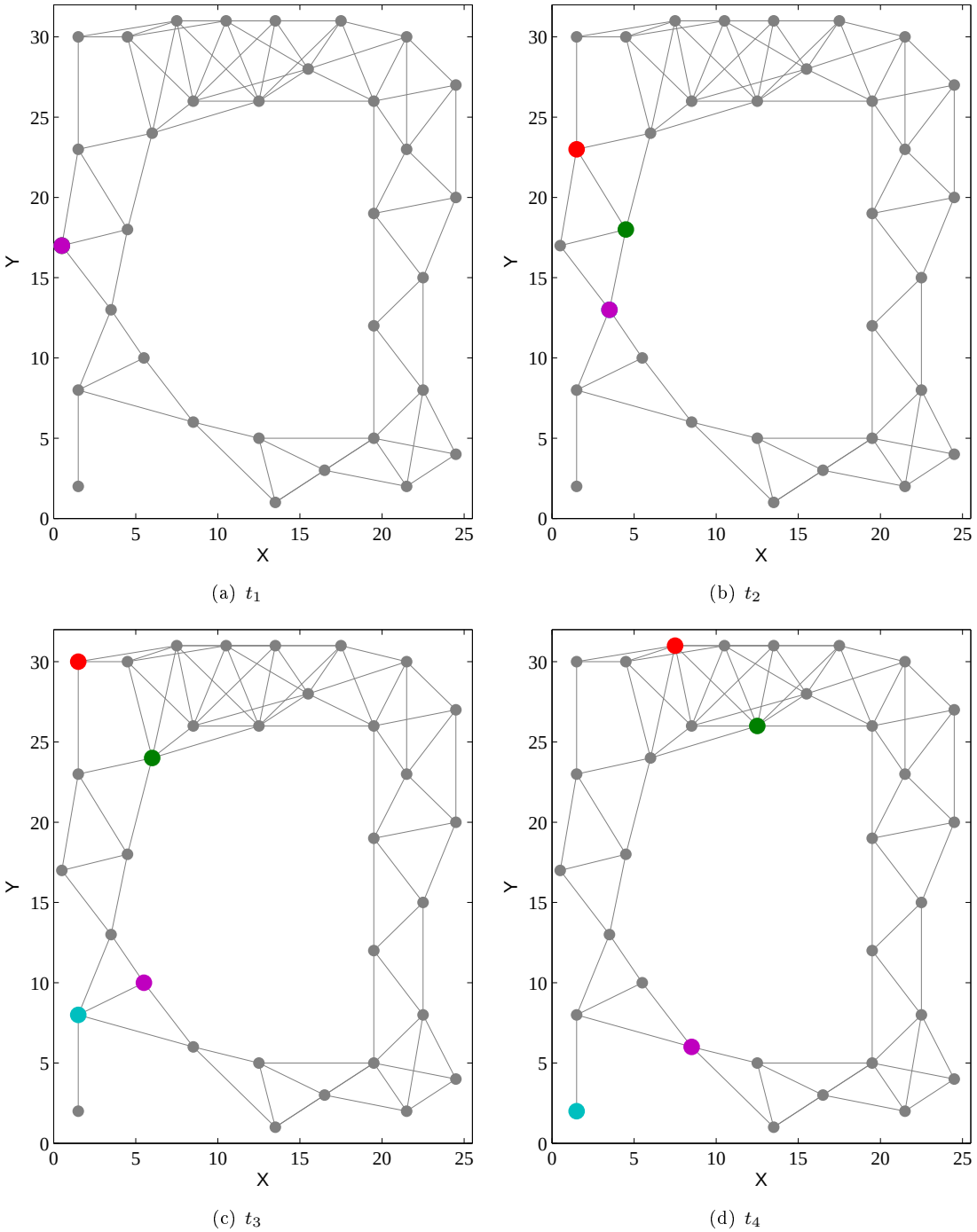
Now, in order to select their movements, sensors need to be able to predict the informativeness (or value) of the samples that are collected along their paths with respect to the accuracy with which measurements at unobserved locations can be predicted. Here, the informativeness of a set of samples that correspond to the random variables $\mathcal{X} = \{\mathcal{X}_1, \mathcal{X}_2, \ldots\}$ is quantified by a function $f(\mathcal{X})$, that, depending on the context, can take on different forms (Meliou et al., 2007). As discussed in Section 2.4, we expressed our preference for the entropy as a metric of informativeness. However, the algorithms presented in the upcoming chapters are not limited to this choice.

To measure the performance of the sensors, we utilise the root mean squared error (RMSE) of their predictions. In order to do this we denote the predictions that the sensors make at time $t$ by $P_t = \{p_t^v | v \in V\}$. Suppose the actual measurements made at those locations are $A_t = \{a_t^v | v \in V\}$, then the objective of the sensors is to coordinate their movements so as to minimise for all timesteps $t \in T$:

$$RMSE(P_t, A_t) = \sqrt{\frac{\sum_{v \in V}(a_t^v - p_t^v)^2}{|V|}} \tag{3.1}$$

Informally, the RMSE can be thought of as a quality measure of the situational awareness the mobile sensors achieve, since it measures the difference between the measurements of the sensors, and the actual values of the phenomena in the environment. We chose this measure because it has been used in related work to ascertain the accuracy of sensor predictions (Guestrin et al., 2005; Krause and Guestrin, 2007).

At this stage of our research, the above formalisation is based on a number of assumptions. As we will discuss later on, the relaxation of these assumptions will be investigated in future work.




(a) $t_1$ (b) $t_2$

(c) $t_3$ (d) $t_4$

FIGURE 3.1: Four timesteps of a team of sensors $\mathcal{S} = \{S_1, S_2, S_3, S_4\}$ moving in an environment whose layout is defined by a graph $\mathcal{G} = (V, E)$, pictured in grey. $E$ contains a pair of locations $(v_i, v_j)$ when they are less than 7.4 meters apart. The initial deployment of the sensors is $\mathcal{L}_1 = (v_1, v_1, v_1, v_1)$, where $v_1 = (0.5, 17) \in V$ (if sensors occupy the same location, only one of them is shown). Phenomenon $\mathcal{P}$ is not shown.

1. There is no cost associated with movement, communication or sensing. Effectively, this implies that sensors have an unlimited power supply.
2. The environment contains a single phenomenon of interest.
3. The team of mobile sensors is homogeneous. Consequently, they have the same movement restrictions, and carry the same type of sensor equipment.

Now that we have a formal definition of the problem, we show in the next section how an *un-negotiated coordination* (see Section 2.5.3) algorithm is capable of tackling it in a distributed way.

# Chapter 4

# Un-negotiated Decentralised Coordination

In the previous chapter, we formalised the mobile sensor monitoring problem and explained its underlying assumptions. In this chapter,[1] we develop a basic decentralised coordination algorithm for a team of mobile sensors. In particular, this algorithm allows the sensors to operate in *un-negotiated coordination* mode. Recall from Section 2.5.3 that this implies that sensors share observations of their environment (belief sharing), but do not explicitly negotiate about their actions; coordination is established exclusively by observing each other's location, and sharing observations from their environment.

Moreover, the algorithm is *greedy*, because at each timestep sensors move to the location with the highest entropy, without considering subsequent moves. This is computationally less demanding than considering the (discounted) information gain of entire paths, because the action space is much smaller when only the next move is considered. Furthermore, the policy is *online*, because the entropy is constantly being updated based on newly acquired observations made by the sensor itself and the other sensors.

This algorithm is inspired by approaches based on techniques found in the literature, such as potential fields, information surfing and myopic information maximisation. The attractiveness of these techniques stems from the fact that they allow sensors to be highly adaptive to their environments: they alternate between updating the model of their environment, and deciding on the next action that maximises information. This is in stark contrast with the offline approaches discussed in Chapter 2 that essentially perform a one-shot optimisation, instead of continuously maximising information during deployment (see Section 2.5.1). Moreover, these approaches are decentralised and enable sensors to make their own decisions, thereby making them autonomous.

[1] The contributions presented in this chapter have been published in Stranders et al. (2008).

However, these methods also have potential drawbacks, originating from the fact that they myopically maximise information. This might result in them getting caught in local maxima, as we will see in Section 4.3, where we empirically evaluate the greedy algorithm. In practical terms, this could seriously impact their ability to tackle complex environments, in which their movement is restricted by obstacles. Finally, these methods are restricted to reasoning about the next move only. In the next chapter, we address this shortcoming and extend the sensors' deliberation cycle to reason about paths.

The remainder of this chapter is organised as follows. In Section 4.1 we present our algorithm in detail, where special attention is paid to its emergent properties. Then, to illustrate these emergent properties, we will show their operation in two example scenarios in Section 4.2. Next, in Section 4.3 we empirically evaluate the coordination algorithm using a real-life dataset. We end the chapter with an overall evaluation of the algorithm in terms of the requirements outlined in Chapter 1, and identify the deficiencies that need to be addressed in future work.

## 4.1 The Coordination Algorithm

In this section, we will present the coordination algorithm based on the problem formalism defined in the previous chapter. Recall from the introduction of Chapter 2, that in order to do adaptive sampling, three challenges have to be addressed. In the remainder of this section, we will highlight each of these challenges in turn. First, we show how we use the GP for information processing. Second, we describe our measure of information. Third, we present the full algorithm for maximising information gain, and minimising the RMSE of the sensor's predictions.

### 4.1.1 Information Processing

In this section, we will explain how phenomena $\mathcal{P}$ is modelled in our approach by a GP. Since $\mathcal{P}$ is a function of time and (two-dimensional) space, we use a GP with three dimensional input vectors: $\mathbf{x} = [\ x \quad y \quad t\ ]^\top$. Now, recall from Section 2.3.2, that a GP with a Matérn covariance function is capable of accurately modelling the properties of the phenomena found in our application domain. For convenience, we repeat it here:

$$k(\mathbf{x}, \mathbf{x}') = \sigma_f^2 \left(1 + \sqrt{3}r\right) \exp\left(-\sqrt{3}r\right) + \sigma_n \delta_{\mathbf{x}\mathbf{x}'} \tag{4.1}$$

where $r \equiv \sqrt{(\mathbf{x} - \mathbf{x}')^\top \mathbf{P}^{-1} (\mathbf{x} - \mathbf{x}')}$, and $\mathbf{P}$ is a diagonal matrix with $\mathbf{P} = diag(l_s^2, l_s^2, l_t^2)$. The diagonal entries of $\mathbf{P}$ are the lengthscales and timescales of phenomena $\mathcal{P}$, and have to be learnt from observations $O$. In order to do this, we employ the techniques outlined in Section 2.3.2.2. More specifically, these techniques allow each sensor to update its GP

model based on the observations $O_t$ that are obtained at each timestep $t$, while at the same time refining the previous estimates of $l_s^2$ and $l_t^2$. In doing so, the GP is able to recover both the temporal and spatial dynamics of the phenomenon. Put differently, it allows the sensors to determine not only the correlation between two measurements over space, but also over time. The significance of this will become clear in the upcoming sections.

### 4.1.2 Value of Information

Based on this GP formulation, we are able to define a value of information in terms of entropic information content of a location. More specifically, if we model a measurement taken at location $v$ as a random variable $X_v$, we can define the entropy of this random variable as the amount of information that can be obtained at location $v$. Now, if we know the realisation $\mathbf{o_{t'<t}}$ of set of observation variables $O_{t'<t}$ taken in the past, we can determine the expected value of information $\mathrm{VI}(v)$ that can be at $v$ as follows:

$$\mathrm{VI}(v) = H(X_v | O_{t'<t} = \mathbf{o_{t'<t}}) \tag{4.2}$$

### 4.1.3 Maximising Information Gain

Given this formulation of information value, the sensors' objective is to choose their movements so as to maximise this information value collectively. Our algorithm that is intended to achieve this is outlined in pseudo code in Algorithm 2. At the beginning of each time step, the sensors take measurements (line 1), and send these to their peers (line 2). Their own measurements, combined with those received from the other sensors in line 3, are used to update the GP model (line 4). Next, each sensor predicts measurements at every location in graph $\mathcal{G}$ for which it is closest (line 6). Finally, in line 8, each sensor chooses the next location $l_{t+1}$ with certain probability $p(l_{t+1})$ proportional to the information value associated with location $l_{t+1}$. More specifically, we use the Boltzmann equation to determine this probability as follows:

$$p(l_{t+1}) = \frac{\exp(\mathrm{VI}(l_{t+1})/\tau)}{\sum_{l_{t+1} \in adj(l_t)} \exp(\mathrm{VI}(l_{t+1})/\tau)} \tag{4.3}$$

Here, $\tau$ is referred to as the temperature variable. It controls the trade-off between maximising utility, and choosing a random action; the higher temperature, the more will the available moves become equally desirable, regardless of their associated utility. When the temperature approaches zero, the selection rule approximates the arg max function. There are two reasons for using this selection rule. The first is that it breaks the symmetry

**Algorithm 2** The coordination algorithm instantiated for sensor $s_i$ at time $t$.

1: Take measurement $o_t^i \leftarrow \mathcal{P}(t, l_t^i)$
2: Send measurement $o_t^i$ and current position $l_t^i$ to sensors $S \setminus s_i$
3: Receive measurements $O_t^{S \setminus s_i}$ and positions $L_t^{S \setminus s_i}$ from other sensors
4: Update GP using newly acquired measurements
5: **for all** $\{v \in \mathcal{V} | \ |l_t^i - v| \leq \left| l_t^j - v \right| \forall j \neq i\}$ **do**
6: Calculate prediction $p_i^v$
7: **end for**
8: Move to adjacent vertex $l_{t+1}^*$ with probability $p(l_{t+1}^*)$ obtained from Equation 4.3.

between the sensors, who would otherwise be inclined to select the same move, since they share the same world view through the exchange of observations. The second reason is that with a sufficiently high temperature, the sensors will move in directions that do not necessarily maximise the information value. Instead, they will move towards areas that might otherwise be left unexplored.

Now, this algorithm has two important emergent properties. The first is the patrolling behaviour of the individual sensors. Using the greedy online entropy policy, sensors will tend to move towards locations with high entropy. Recall that the covariance function we use not only takes into account the spatial correlations in the environment, but also the temporal correlations. As a result, observations made in the past will have increasingly less relevance for predicting the current state of the environment. The entropy at locations that remain unvisited will therefore increase automatically, and those locations consequently become increasingly attractive to visit again. This effectively incentivises the sensors to be in a constant state of hill climbing in the direction of the steepest entropy gradient (cf. *information surfing*, and the potential field technique discussed in Section 2.5.2). So, it is crucial for the mobile sensors to model the temporal dynamics of the spatial phenomena, because the problem they face is not a one-off optimisation problem; the sensors need to determine the next informative placement given newly acquired observations, while the relevance of older observations declines over time. Consequently, they will have to keep patrolling the environment by revisiting previously visited locations.

The second emergent property is the coordination between the sensors. The exchange of observations in step 2 of the algorithm allows the sensors to share the same world view. It enables them to calculate the entropy for every possible move, based on their neighbours' locations and observations. Sensors will therefore tend to avoid each other, and spread out, because moving in the direction of another sensor will generally decrease the entropy. As a result, the sensors are capable of implicitly coordinating their actions through the exchange of simple observations, and need not also exchange their plans or intentions.

In the next section, we illustrate the operation of this algorithm with a number of example scenarios.

## 4.2 Example Scenarios

In this section we will present two example scenarios that illustrate the emergent properties discussed above. First, however, we will briefly discuss the simulator that we have developed for the purpose of evaluating different sensor strategies. Next, we present the first scenario, which features a single sensor that is given the task of monitoring an environment. This scenario is intended to illustrate the sensor's tendency to be attracted to areas of higher uncertainty. In the second scenario, two sensors are monitoring a slightly more complex environment. In this scenario, we will see an example of emergent coordination between two mobile sensors. At the end of the section, we will give a link to the videos we have created of some of our simulations, in order to give a more dynamic view of the operation of the algorithm.

### 4.2.1 The Simulator

In order to empirically evaluate our coordination algorithm, we have built a simulator for mobile sensors in JAVA. This simulator contains an implementation of the GP, the learning algorithms presented in Section 2.3.2.2 and the problem formalisation that we developed in Chapter 3. As such, it has enabled us to test the effect of different parameter settings and visualise the behaviour of our mobile sensors in different environments. Moreover, its architecture has been specifically designed with future requirements in mind, and it is therefore expected that future extensions are straightforward to implement.

### 4.2.2 Scenario 1: A Single Sensor

This scenario features a single sensor moving through a simulated version of the Intel Berkeley Lab. This lab contains a fixed WSN of 54 sensors, and a dataset collected by this WSN is publicly available on the Internet.[2] This dataset has been used extensively in related work to evaluate adaptive sampling algorithms (e.g. Guestrin et al. (2005)), as well as in our empirical evaluation.

Now, Figure 4.1 shows a snapshot of the algorithm in action. It shows the approximate path of single sensor, and the variance with which the sensor is able to predict measurements throughout the environment at time $t = 25$. As can be seen, the sensor is moving in the direction of higher variance (and thus entropy). Note how the variance behind the sensor increases along the travelled path. This illustrates the effect of modelling the phenomenon's time dynamics: the older the measurement at a certain location, the less useful it is to predict the current measurement.

[2] Refer to `http://db.csail.mit.edu/labdata/labdata.html` for more details.

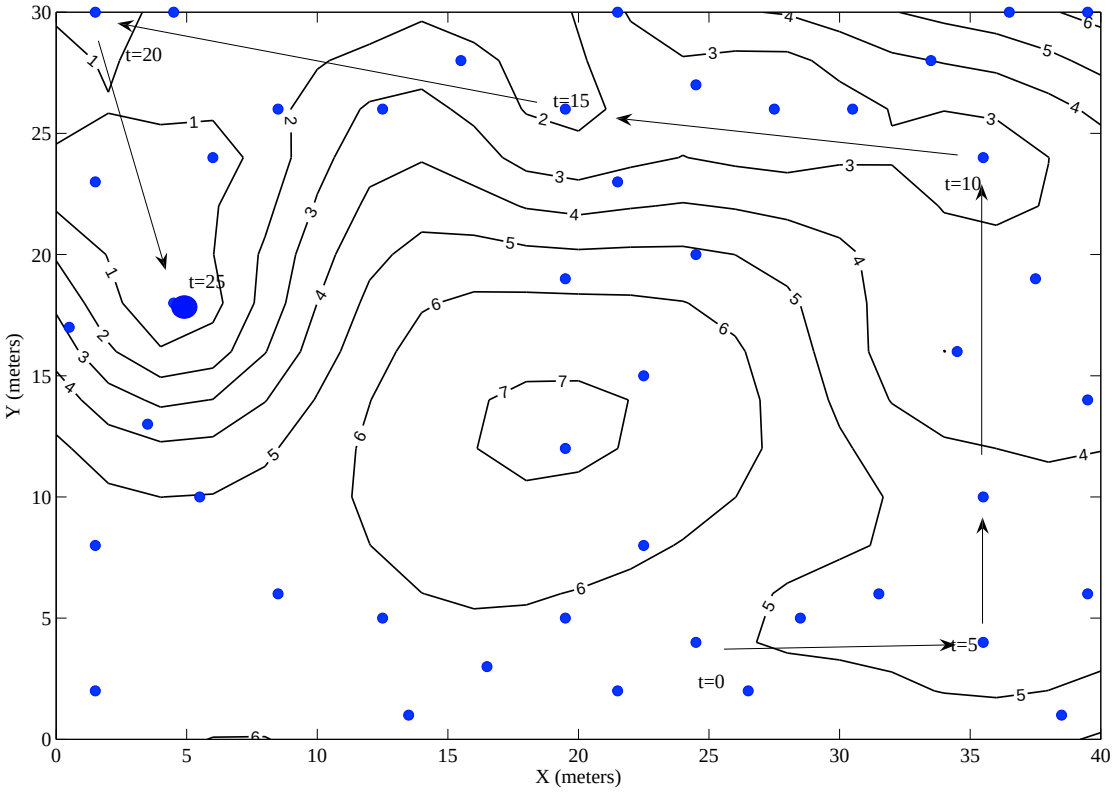



FIGURE 4.1: The world view of a single sensor moving through the Intel Berkeley Research Lab at timestep $t = 25$. The blue dots represent the locations $V$ the sensor can move between, and the arrows indicate the approximate path of the sensor. Superimposed is a contour plot of the predictive variance with which the sensor can predict measurements throughout the environment (the numbers on the contour lines indicate the variance). The lower the predictive variance, the better the measurements at those locations can be predicted. These predictions are made using the sensor's GP, that is trained with the measurements collected along the sensor's path. Times are in minutes.

### 4.2.3 Scenario 2: Coordination between Two Sensors

In this scenario, two sensors are exploring an environment that consists of a long corridor that is connected to a large room. Figure 4.2 shows four snapshots of this scenario, taken at timesteps 100, 200, 300, and 400. That is to say, the sensors have made 100 moves and observations in the interval between two snapshots. From this figure, we can observe that in the first 200 timesteps, the coordination algorithm allocates the green sensor to the corridor, and the blue sensor to the large room. In doing so, the sensors are capable of monitoring both the corridor and the room at the same time. Even more interesting is the green sensor's behaviour between timesteps 200 and 300. Having sufficiently reduced the uncertainty in the corridor, it decides to move into the large room to assist the blue sensor. A comparison of the predictive variance (or uncertainty) of the snapshot at timesteps 200 and 300 shows the effectiveness of this strategy; the uncertainty in the room is reduced. Also, we can see that the uncertainty in the corridor has increased, as a result of the fact that it is no longer covered by the green sensor. By the end of timestep

400, the environmental conditions within the corridor are again maximally uncertain, as if no sensor has patrolled it before. However, this uncertainty attracts the green sensor, which is about to enter the corridor once again. In sum, the behaviour that the sensors exhibit in this scenario shows the effectiveness of the emergent coordination between the sensors.

### 4.2.4 Simulation Videos

We created two videos of these two simulations, which can be found at `http://users.ecs.soton.ac.uk/rs06r/videos/`. Unlike the static snapshots in Figures 4.1 and 4.2, these videos offer a more dynamic view of the operation of the algorithm.

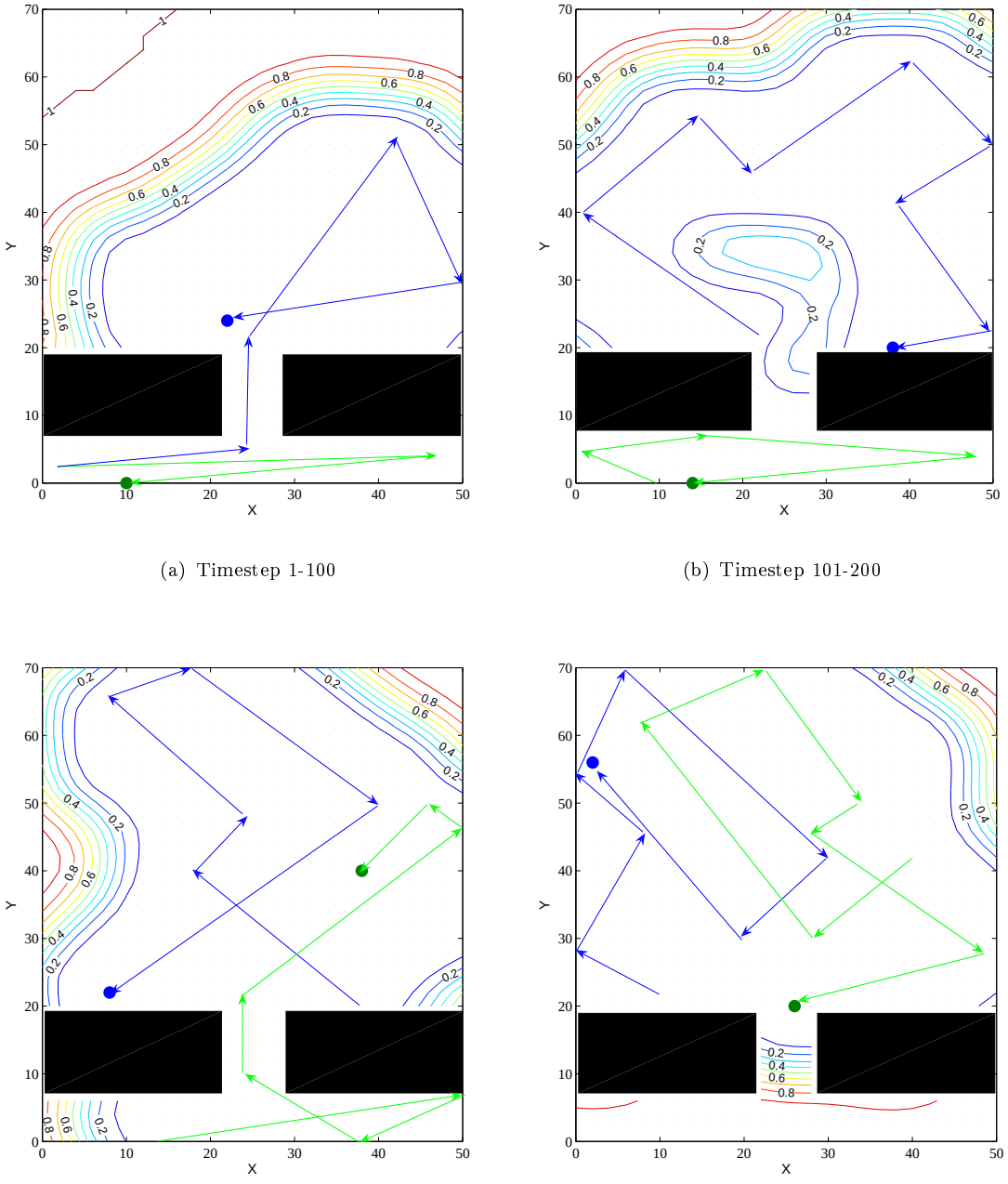
## 4.3 Empirical Evaluation

The scenarios discussed in the previous section give a feel of the way in which the sensors behave with our algorithm. However, this provides only a qualitative analysis of the algorithm, and does not give an indication of its performance in terms of the quantitative measures defined in Chapter 3. This section is therefore dedicated to a more thorough empirical evaluation. This section is organised as follows. First, we will explain the experimental setup we used. Next, we present and analyse the results.

### 4.3.1 Experimental Setup

To empirically evaluate our approach, we used our simulator to compare the performance of different teams of 5 sensors using a dataset from the Intel Berkeley Research Lab. The dataset contains temperature, humidity and light intensity measurements collected by 54 fixed wireless sensors between 28 February and 5 April 2004. Figure 4.3 shows the layout of the lab, and the location of the sensors.

We compared our approach with five benchmark policies using this dataset. To do this, we first instantiate our formal model introduced earlier for this environment. More specifically, the graph $\mathcal{G}$ is populated with vertices $V$ that correspond to the 54 sensor locations, and edges $E$ that restrict movement between any pair of locations $v_i, v_j \in V$ no further than 8 meters apart. At every 5 minute interval, each deployed sensor $s_i$ makes a temperature reading $o_t^i$ by querying the Berkeley dataset for the sensor's present location $l_t^i$ and the current simulation time $t$. After the sensors have updated their GP models (Algorithm 2, step 4), the sensors collectively make temperature predictions $P_t$ for all of the 54 sensor locations $V$ (step 6). Next, we determine the accuracy of these predictions in terms of the RMSE by comparing them to the actual readings $A_t$. Finally, the policies that use mobile sensors decide where to move next.




(a) Timestep 1-100

(b) Timestep 101-200

(c) Timestep 201-300

(d) Timestep 301-400

FIGURE 4.2: Snapshots at timesteps 100, 200, 300 and 400 of two sensors moving through an environment with a corridor connected to a large room. The grid points represent the locations the sensors can move between, and the arrows indicate the approximate paths of the sensors in the 100 timesteps between two subsequent snapshots.

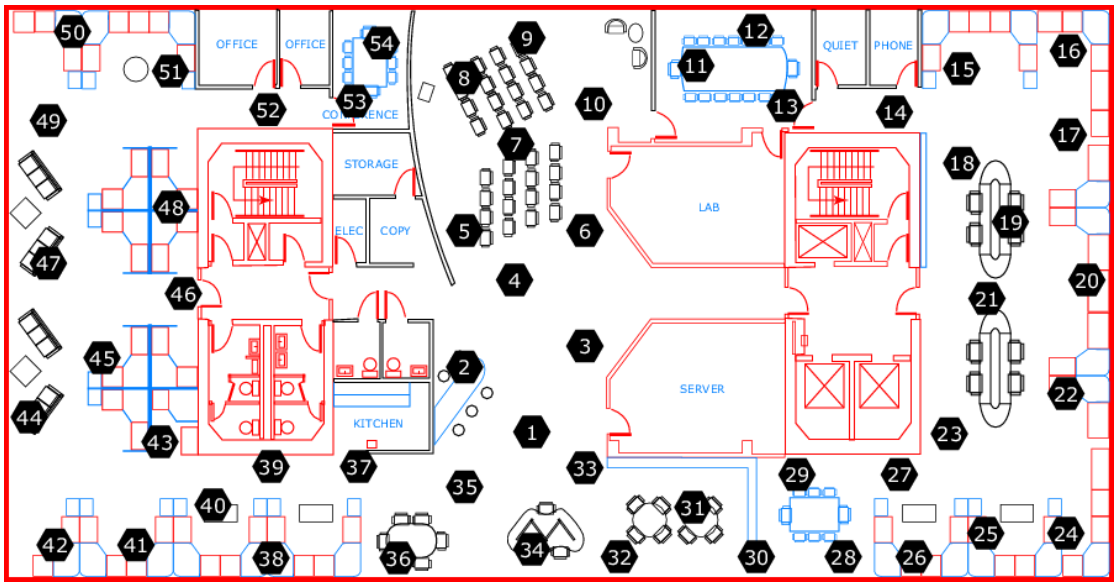



FIGURE 4.3: Sensor deployment at the Intel Berkeley Research Lab. In our simulation, the sensors can move between the 54 real sensor-locations every 5 minutes to a location within 8 meters of their previous position. The lab itself measures 30 by 40 meters. (Copied from `http://db.csail.mit.edu/labdata/labdata.html`).

Now, the six policies used in this experiment can be seen as points in three-dimensional policy space: the first dimension specifies the movement: mobile (M), jumping (J), and fixed (F). The second dimension describes learning: knowing (K) the hyperparameters in advance[3], or having to learn (L) them. The last dimension specifies the type of policy: greedy (G) or random (R). In case of greedy, it is also specified what is greedily maximised: entropy (e) or mutual information (mi).

**MLGe** Our algorithm as detailed in the previous section: mobile sensors that are capable of learning the values of the hyperparameters using BMC, and employ a greedy online movement policy for entropy maximisation.

**MLR** The same mobile sensors as MLGe, except that they move randomly. This policy was included to determine the effect of the greedy entropy policy.

**MKGe** The same sensors as MLGe, except that they have prior knowledge of the all hyperparameters. This policy evaluates the effect of having to learn the hyperparameters.

**JKGe** The same as MKGe, except that these sensors can instantaneously jump to a desired location without visiting intermediate locations. This policy acts as an upper bound for achievable performance.

[3]Using an offline learning algorithm, we determined the values of the hyperparameters of the temperature in the lab.

**FKGmi** Fixed sensors that are placed using a greedy mutual information maximisation algorithm that has prior knowledge of all the hyperparameters (Guestrin et al., 2005).

**FKGe** Fixed sensors that are placed using a greedy entropy maximisation algorithm that has prior knowledge of the hyperparameters.

Given the characteristics of these policies, we formulate the following experimental hypothesis:

> *Hypothesis* 1. The prediction accuracy of our algorithm (MLGe) will be higher than that of both fixed sensor networks, as well as that of the random policy. The former is a result of the fact that the mobile sensors controlled by our algorithm are able to collect observations from the entire environment, and are not restricted to sampling at locations that have been determined at design time. The latter is caused by the fact that a random policy will not select observations based on their information value, and will therefore be less successful in reducing the uncertainty in the sensors' models. As a result, the predictions of these models are less accurate.
>
> *Hypothesis* 2. Our algorithm will closely approximate the performance of benchmarks MKGe and JKGe that possess prior knowledge of all the hyperparameters, that our algorithm has to learn online. This is due to the efficiency of the learning algorithm.

### 4.3.2 Results

Figure 4.4 shows the accuracy of the predictions in terms of RMSE of a single run through the dataset, averaged over the days on which the measurements were made. It clearly shows that our mobile sensors outperform both fixed placements, and the randomly moving sensors, confirming Hypothesis 1. Furthermore, the fact that the randomly moving sensors perform worse clearly shows that the greedy entropy policy is indeed effective. Finally, the prediction accuracy of our mobile sensors is comparable to those sensors that can instantaneously jump to their desired location, and the mobile sensors that have prior knowledge of the hyperparameters (i.e. the two policies used to put an upper bound on achievable performance), confirming Hypothesis 2.

Figure 4.5 shows the effect of varying the number of sensors in the environment. The mobile sensors start off with a significantly higher performance than the fixed sensor network; around 15 fixed sensors with prior knowledge of the hyperparameters are needed to attain the same performance as a team of 5 mobile sensors. However, when the number of sensors increases to around 15 (30% of the number of sensors in the Berkeley Lab), the two solutions become equivalent. At this point, increasing the number of

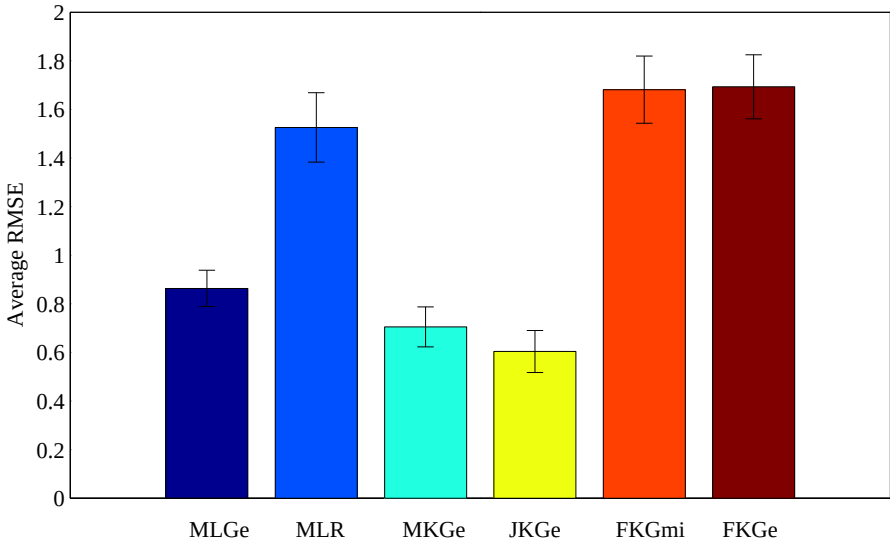



FIGURE 4.4: Average RMSE for the different types of sensor policies. Each simulation was performed with 5 sensors. The error bars indicate the standard error of the mean.

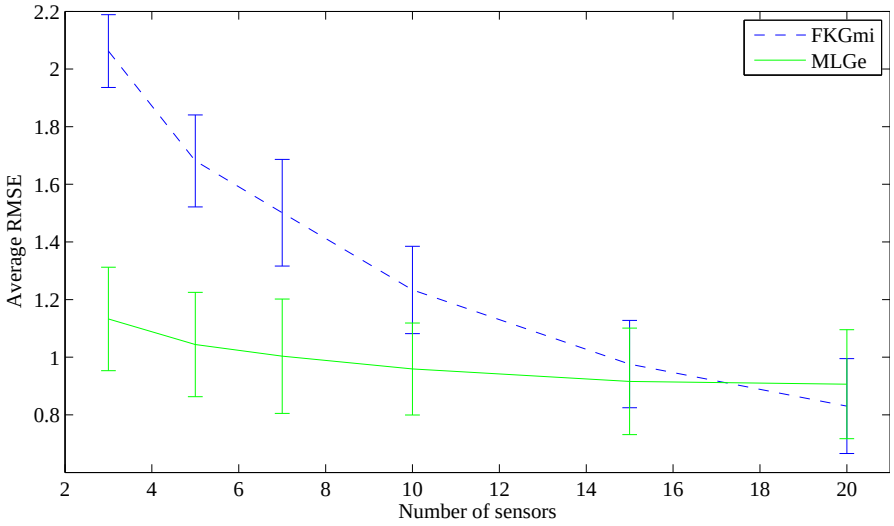



FIGURE 4.5: Graph showing the effect of varying the number of sensors in the environment. Around 15 fixed sensors are needed to attain a similar performance to the 5 mobile sensors. Additionally, the graph shows that adding additional moving sensors to the environment does not bring about a significant increase in prediction quality. The error bars indicate the standard error of the mean.

fixed sensors has a greater effect on the performance than introducing additional mobile sensors. This is caused by the fact that as the number of mobile sensors increases, the freedom of movement of the sensors is reduced, thereby making it more difficult for them to reach locations of high entropy. As a result, the performance of the mobile sensors increases more slowly. We believe that this can be prevented by introducing more explicit coordination between the sensors, which is considered in future work.

## 4.4 Summary

In this chapter, we showed our first contribution to the field of mobile sensing. Specifically, we introduced a decentralised algorithm for coordinating teams of mobile robots for monitoring environmental phenomena in dynamic environments. With this algorithm, sensors are able to learn the hyperparameters of the GP that models these phenomena online, and coordinate their actions to maximise the collective information gain. At this point, we will review this algorithm in terms of the requirements laid down in Chapter 1, and identify the deficiencies that will be addressed in future work.

**Accuracy:** The GP regression techniques used within the algorithm are able to recover the spatial and temporal dynamics of an uncertain environment. However, so far, we have modelled these dynamics with a stationary covariance function, which exhibits the same properties at each spatio-temporal coordinate within the environment. In practical settings, however, some areas of the environment might be more dynamic than others, necessitating the use of a non-stationary covariance function. We intend to identify efficient and practical techniques in future work.

**Relevance:** Our current value of information does not distinguish between relevant or irrelevant events. It places value on an observation solely based on the grounds of its potential to reduce uncertainty in the sensor's model. The notion of relevance should therefore be incorporated in the measure of information. This way, sensors will be incentivised to reduce uncertainty in the environment and at the same time collect those observations that are relevant in terms of the decision maker's goals.

**Adaptiveness:** The GP regression techniques combined with the potential to learn the hyperparameters of the covariance function make the sensors highly adaptive to their environment. However, there is still an open question of how adaptive GP regression for non-stationary processes should be performed.

**Robustness:** The current algorithm is robust against the failure of a single sensors, since it does not contain any centralised elements. Moreover, we can conclude from Figure 4.5 that its performance will degrade gracefully in the face of multiple sensor failures. However, the robustness of the algorithm in the presence of unreliable communication links needs still to be investigated.

**Autonomy:** With this algorithm, the sensors are in full control of their own actions. However, in the current approach, each sensor has access to all observations from all sensors, resulting in identical world views. In contrast, in more realistic scenarios, where the communication bandwidth and range are limited, this will no longer be feasible. As a result, the sensors' world models will start to diverge. We will need to investigate how sensors should cope with this, by making decisions based only on information that is available through their immediate neighbours and their own observations.

**Scalability:** The current communication protocol requires the sensors to exchange all their observations with each other. Effectively, this means that when adding an additional sensor to an existing team of $n$ sensors, $2(n+1)$ additional messages need to be sent. As a result, the number of communicated messages will increase quadratically with the number of sensors. While this problem might be alleviated if we allow for broadcasts, we believe that the this property should be further investigated. Potential solutions include restricting the communication to immediate neighbours only, or increasing the level of abstraction of the contents of the messages by aggregating multiple observations into a single high-level representation.

**Modularity:** In the experiments, we have modelled the sensors as being identical. However, the formalism in Chapter 3 can easily be adapted to allow for different movement capabilities of the sensors. Moreover, the communication protocol requires that the sensors exchange raw observations, which does not impose many restrictions on the implementation of the types of sensors used.

The algorithm presented in this chapter allows mobile sensors to operate in un-negotiated cooperation mode. Thus, the sensors share the same world-view through the exchange of observations, but do not negotiate about their collective action. In the next chapter, we investigate and evaluate the a *negotiated coordination* algorithm (see Section 2.5.3). As will become evident, this algorithm has the potential of making the mobile sensors more robust in complex environments, by preventing the sensors from getting trapped in local minima.

# Chapter 5

# Negotiated Decentralised Coordination

In the previous chapter, we developed a decentralised coordination algorithm that operates in *un-negotiated* mode; it allows mobile sensors to coordinate exclusively by sharing observations of their environment. No explicit negotiation is required to determine their next moves; sensors move according to what they believe obtains the most valuable samples, regardless of how their neighbours decide to reposition themselves. Put differently, sensors attempt to maximise their own immediate utility, without taking into consideration the impact their decision might have on other sensors' utility. As we will see in this chapter,[1] a more coordinated approach, in which the sensors explicitly negotiate about their next joint move *prior* to executing it, can lead to significantly improved performance. More specifically, the algorithm that we develop in this chapter uses *negotiated coordination* (see Section 2.5.3). In order to accomplish this, we will drop the rather individualistic approach of the previous chapter in favour of a more collective perspective on the behaviour of the sensors. Most importantly, we shall do this without losing the desirable properties of scalability, robustness, modularity, and autonomy.

In more detail, the negotiated coordination algorithm we present in this chapter attempts to maximise the information gain obtained by the team as a whole. To see why this can lead to improved performance compared to the un-negotiated coordination algorithm, note that the latter controls sensors individually to make locally greedy decisions. These decisions create constraints for other sensors to deal with, which are likely to force the algorithm into a local maximum. As a result, their behaviour can result in strongly sub-optimal paths through their environment. In contrast, a negotiated coordination algorithm can avoid these local maxima, by resolving conflicts of interests between the sensors before any decision is made.

[1] This chapter is based on Stranders et al. (2009b).

Additionally, instead of planning just a single move, the negotiated algorithm considers the action space of the entire team, which makes it possible to extend the planning horizon and consider paths of longer length. This leads to increased performance, since a more distant planning horizon allows sensors to take into account the effects of their moves further into the future.[2] The un-negotiated algorithm was not capable of doing this, because it required sensors to observe their neighbours' positions in order to determine their next move. Planning ahead further than a single move would cause the sensors to lose synchronisation, causing them to base their decisions on stale information, which would further decrease their performance.

Given this background, the remainder of this chapter is organised as follows. First, we present the negotiated coordination algorithm. Second, we illustrate the operation of the algorithm with an example scenario. Third, we empirically evaluate the new algorithm, and demonstrate its effectiveness. In particular, we show that the algorithm substantially outperforms the greedy un-negotiated algorithm that was developed in the previous chapter.

## 5.1 The Coordination Algorithm

Similar to the un-negotiated coordination algorithm from the previous chapter, the algorithm we develop in this chapter addresses the three challenges of adaptive sampling (see Chapter 2): information processing, information valuing, and maximising information.

Now, since this algorithm makes use of the Gaussian process in the same way as the algorithm from previous chapter, we do not address the information processing aspect of this algorithm. Instead, we focus on the two remaining challenges. In more detail, we first define the value of information that a single sensor receives in terms of its contribution to the collective information gain of the team. Second, we show how the joint action space is defined for multiple sensors and how this enables the sensors to plan paths of arbitrary length. Also, we show how the max-sum message passing algorithm for decentralised coordination can be applied to compute the joint action that maximises the information gain for the team as a whole.

### 5.1.1 Value of Information

As mentioned earlier, the algorithm that we develop in this chapter considers the performance of monitoring the spatial phenomenon of the *entire team*, instead of the performance of single individual sensors. Recall from Section 4.1.2 that performance is

[2] However, unlike the offline algorithms discussed in Chapter 2, this online decentralised algorithm allows for paths to be changed whenever unexpected events occur, such as the failure of a sensor, or the discovery of a rapidly changing phenomenon in the environment.

measured in terms of entropy reduction. Now, instead of determining the entropy reduction caused by a single observation, as was done in the un-negotiated case, we now have to determine the entropy reduction caused by a set of observations collected by the team of sensors. More formally, the $M$ sensors need to determine the value of collecting observation sets $A_1, \ldots, A_M$, where $A_i$ is collected by sensor $i$, relative to the observations $B$ that were collected previously. This value is equal to $H(A_1 \cup \ldots \cup A_M | B)$, or the conditional entropy of the random variables corresponding to the observations $A_i$, conditioned on the observations $B$. Trivially, this value can be computed by each sensor if it knows all observation sets $A_1, \ldots, A_M$. This, however, would imply that each sensor has to consider (and thus have knowledge of) all observations of all sensors. Needless to say, this would lead to a large communication overhead. What is needed is a more localised and decentralised algorithm for computing the value of observations $A_1, \ldots, A_M$.

A naïve approach—that, as it turns out, is incorrect—would be to sum the entropies of the samples each individual sensor collects. In this way, sensors do not need to share their observations, and only need to maximise the value of the observations they themselves decide to collect. Note however, that this method overestimates the true global utility. In more detail, $H(A_i) + H(A_j) \geq H(A_i \cup A_j)$, with equality only if the sets of random variables $A_i$ and $A_j$ are independent.[3] So, in general, $\sum_{i=1}^{M} H(A_i) \neq H(A_1, \ldots, A_M)$. Consequently, if each sensor individually attempts to maximise its utility by collecting samples of high value, the sensors collectively will generally not maximise the team utility.

However, the following theorem gives rise to a more sophisticated way of defining team utility. First, we define the key concept of *incremental value* $\rho$:

**Definition 5.1.** The *incremental value* of adding a set of observations $A = \{\mathcal{X}_1, \mathcal{X}_2, \ldots\}$ to another set of observations $B = \{\mathcal{X}'_1, \mathcal{X}'_2, \ldots\}$ is the information gain obtained by collecting $A$, and is denoted as $\rho_A(B) = H(A \cup B) - H(B)$.

**Theorem 5.2.** *Let* $\{A_1, \ldots, A_n\}$ *be a partition of* $A$*. Then,*

$$H(A|B) = \sum_{i=1}^{n} \rho_{A_i} \left( \bigcup_{j=1}^{i-1} A_j \cup B \right)$$

[3] In most environments, and through the property of *locality*, two sets of observations can be considered independent if they have been collected "far" apart. The exact definition of "far" depends on the length scales of the spatial phenomenon that is monitored (see Section 2.3.2).

*Proof.*

$$\begin{aligned} H(A|B) &= H(A_1|B) + H(A_2|A_1, B) + \ldots + H(A_n|A_1, \ldots, A_{n-1}, B) \\ &= [H(A_1 \cup B) - H(B)] + [H(A_1 \cup A_2 \cup B) - H(A_1 \cup B)] + \ldots + \\ &\quad [H(A_1 \cup \ldots \cup A_n \cup B) - H(A_1 \cup \ldots \cup A_{n-1} \cup B)] \\ &= \rho_{A_1}(B) + \rho_{A_2}(A_1 \cup B) + \ldots + \rho_{A_n}(A_1 \cup \ldots \cup A_{n-1} \cup B) \\ &= \sum_{i=1}^{n} \rho_{A_i}\left(\bigcup_{j=1}^{i-1} A_j \cup B\right) \end{aligned}$$

□

Here, the first equality is obtained by repeatedly applying the chain rule of entropies: $H(X, Y) = H(X|Y) + H(Y)$.

Informally, Theorem 5.2 tells us that the team utility is a sum of the incremental values by adding samples $A_i$ to the samples collected by sensors $j < i$. We will call the individual factors of this sum the *sensor contributions*, or *sensor utility*.

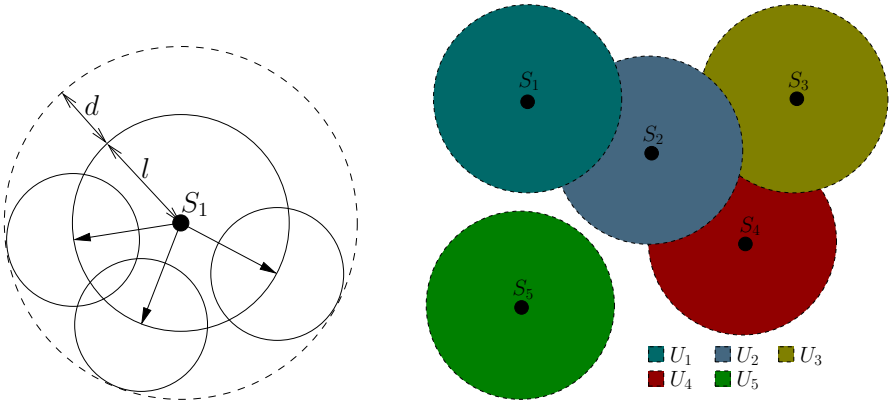
**Definition 5.3** (Sensor Contribution/Utility)**.** Sensor $i$'s contribution to the team utility is:

$$U_i(A_1, \ldots, A_i) = \rho_{A_i}\left(\bigcup_{j=1}^{i-1} A_j \cup B\right)$$

In order to calculate its contribution, a sensor need only be aware of the samples collected by sensors with a lower ID. Moreover, by summing the contributions by individual sensors, we obtain $H(A|B) = H(A_1 \cup \ldots \cup A_M|B)$, which is the team utility. As we will see in the upcoming sections, these two properties are necessary requirements for applying the max-sum algorithm, and can be exploited to obtain an intuitive and efficient model of computation and communication.

Further simplifications to the result of Theorem 5.2 can be made if we take into account that certain sets of variables are independent. For example, if the variables in the set $A_k$ are independent from the variables in the sets $\{A_1, \ldots, A_i\} \setminus A_k$, then:

$$\begin{aligned} \rho_{A_i}\left(\bigcup_{j=1}^{i-1} A_j \cup B\right) &= H(A_1 \cup \ldots \cup A_i \cup B) - H(A_1 \cup \ldots \cup A_{i-1} \cup B) \\ &= [H(A_1 \cup \ldots \cup A_{k-1} \cup A_{k+1} \cup \ldots \cup A_i \cup B) + H(A_k)] - \\ &\quad [H(A_1 \cup \ldots \cup A_{k-1} \cup A_{k+1} \cup \ldots \cup A_{i-1} \cup B) + H(A_k)] \\ &= \rho_{A_i}\left(\bigcup_{j=1, j\neq k}^{i-1} A_j \cup B\right) \end{aligned} \tag{5.1}$$




(a) Influence circle of a single sensor (explanation in text).

(b) An example scenario with 5 sensors.

FIGURE 5.1: The *influence circles* of mobile sensors.

When applying the Gaussian process in the way we discussed in Section 2.3.2, two sets of observations are never fully independent. This results from the fact that the covariance functions we chose to use assign non-zero covariance to every pair of observations (random variables). However, if $X$ and $Y$ are almost independent, meaning that collecting observations $X$ gives only very little information about (potential) observations $Y$, then $H(X,Y) = H(X) + H(Y) - \epsilon$ (with $\epsilon$ very small), in which case independence is a reasonable approximation.

For most spatial phenomena encountered in reality, there exists a maximum distance $d$ over which the act of collecting a sample reduces the entropy by more than $\epsilon$. If $l$ is the length of the path over which negotiation is taking place, then samples collected outside their *influence circle* with radius $d + l$ centred at the sensor's current location can be considered independent of a sample collected within the circle with radius $l$ (see Figure 5.1(a)).

To clarify this issue further, the following example illustrates the ideas developed in this section.

**Example 5.1.** *Depicted in Figure 5.1(b) are 5 sensors with their respective influence circles. Table 5.1 summarises the contribution functions $U_i$ of each sensor to the team utility. Using Equation 5.1, the expression for their contribution functions can be further simplified, based on whether the sensors' influence circles overlap.*

So far, we have defined how team utility is factored into the individual sensors' contributions. For sensor $i$, this contribution is calculated by determining the *incremental value* of adding observations $A_i$ to the set of observations collected by sensors $j < i$, and the observations that were collected previously. However, the observations in $A_i$ are subject to various constraints, that we have not yet considered. Most importantly,

| Sensor | $U_i$ | Simplified $U_i$ |
|---|---|---|
| $S_1$ | $\rho_{A_1}(B)$ | Simplification not possible |
| $S_2$ | $\rho_{A_2}(A_1 \cup B)$ | Simplification not possible |
| $S_3$ | $\rho_{A_3}(A_1 \cup A_2 \cup B)$ | $\rho_{A_3}(A_2 \cup B)$ |
| $S_4$ | $\rho_{A_4}(A_1 \cup A_2 \cup A_3 \cup B)$ | $\rho_{A_4}(A_2 \cup A_3 \cup B)$ |
| $S_5$ | $\rho_{A_5}(A_1 \cup A_2 \cup A_3 \cup A_4 \cup B)$ | $\rho_{A_5}(B)$ |

TABLE 5.1: Contribution functions for the scenario in Figure 5.1(b).

$A_i$ can only contain those observations that are collected along a *path* from sensor $i$'s current location. These constraints are the focus of the next section, where we define the action space of individual sensors, the joint action space of the team as a whole, and the decentralised coordination algorithm to maximise information value in the presence of these constraints.

### 5.1.2 Maximising Information Gain

Having formulated the value of information contributed by a single sensor, we now turn to the challenge of maximising it in the presence of the movement constraints that were defined in Chapter 3. In more detail, we first discuss the movement constraints of the sensor team, and the joint action space of the team. Second, we define the objective function that the team should maximise. Third, we show that the max-sum algorithm for decentralised coordination (Farinelli et al., 2008) can be used to achieve this in a decentralised fashion. Fourth, we present two novel generic pruning algorithms that are aimed at speeding up the max-sum algorithm in order to reduce the computational overhead incurred by this algorithm computationally and to make it possible to solve the mobile sensor monitoring problem in a timely fashion. Fifth and finally, we develop a technique for incentivising the team of sensors to maintain network connectivity.

#### 5.1.2.1 The Action Model

Recall from the problem formulation in Chapter 3 that the motion of the mobile sensors is constrained by a graph $\mathcal{G}$ that defines the layout of the sensors' environment. Observations can only be collected at the vertices of $\mathcal{G}$, and moves between two vertices are only allowed if this graph contains an edge between them. As a result, the observations $A_i$ that sensor $i$ can collect are restricted by its current location, the layout of the environment, and the maximum length $l$ of the path over which coordination takes place.

So, given a sensor's current location $v_i$, and path length $l$, the set of all possible paths that the sensor can currently consider is a set which we will denote by $\mathcal{A}_i$. The cardinality of this set depends on $\mathcal{G}$ and $v_i$, and is a non-decreasing function of $l$. The joint action space

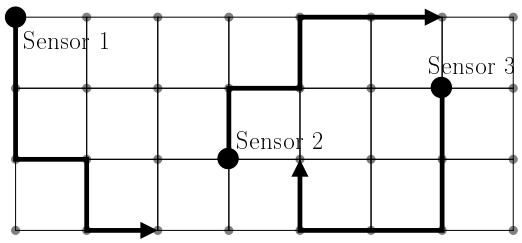



FIGURE 5.2: A joint move of length 5 for sensors on a lattice graph.

$\mathcal{A}$ of the team of sensors is then the Cartesian product of all individual action spaces: $\mathcal{A} = \times_{i=1}^{M} \mathcal{A}_i$. Note that this grows exponentially with the number of sensors $M$, as well as with the average cardinality of the action spaces of individual sensors. Therefore, it is prudent to make the individual action spaces $\mathcal{A}_i$ as small as possible, since computation scales proportionally to the number of possible joint actions. Moreover, many actions in $\mathcal{A}_i$ are unlikely to result to an optimal solution (for instance, a move that makes a sensor visit the same location twice in a short period of time). Furthermore, the sensors might only consider those moves that keep them within their own partition of the environment, or those moves that keep them within communication range of one another.

Now, each element $\mathbf{p} \in \mathcal{A}$ is a collection of $M$ paths of length $l$; one for each sensor. An example of such a joint move for three sensors, consisting of a path of length 5 for each of them, is shown in Figure 5.2. A joint move is an ordered list of vertices of $\mathcal{G}$, at each of which the sensor makes an observation of the spatial phenomenon. Thus, there exists a correspondence between a path and a collection of observations. Therefore, with some abuse of notation, we can treat every element in $\mathcal{A}_i$ as both a path and a set of observations. We have now established a correspondence between an element $\mathbf{p} = \{v^{(1)}, v^{(2)}, \dots\} \in \mathcal{A}_i$, and a set of observations $A_i$, as defined in the previous section.

More formally, we assign to each sensor $i$ a decision variable $p_i$, which takes values in the set $\mathcal{A}_i$, representing all moves that sensor $i$ is currently considering. Applying definition 5.3, we know that the *contribution* of sensor $i$ to the team utility is equal to $U_i(p_1, \dots, p_i)$.

At this point, we have defined the action space of the team of sensors, and the contribution functions that assign a value to each of these actions. What remains is an algorithm that attempts to find the joint action that maximises team utility.

#### 5.1.2.2 The Objective Function

The central problem is now to find observation sets $A_1^*, \dots, A_M^*$ that maximise the conditional entropy given observations $B$ that were made previously:

$$\{A_1^*, \dots, A_M^*\} = \underset{A_1,\dots,A_M}{\arg\max}\, H(A_1 \cup \dots \cup A_M | B) \tag{5.2}$$

Using Theorem 5.2, and Definition 5.3, we can transform Equation 5.2 into a sum of sensor contributions:

$$\{A_1^*, \ldots, A_M^*\} = \arg\max_{A_1,\ldots,A_M} \sum_{i=1}^{M} U_i(A_1, \ldots, A_i) \tag{5.3}$$

or, by applying the movement constraints imposed by $\mathcal{G}$, and using the correspondence between decision variables $p_i$ and observation sets $A_i$ established in the previous section:

$$\mathbf{p}^* = [p_1^*, \ldots, p_M^*] = \arg\max_{p_1 \in \mathcal{A}_i, \ldots, p_M \in \mathcal{A}_M} \sum_{i=1}^{M} U_i(p_1, \ldots, p_i) \tag{5.4}$$

In other words, the sensors attempt to find joint move $\mathbf{p}^*$ that maximises the team utility by maximising the sum of their contributions as defined in Definition 5.3.

Now, it is important to note that we can simplify this equation by exploiting the fact that some observations are (almost) independent, by applying Equation 5.1. As a result, the number of parameters of the utility functions $U_i$ can be reduced; that is, some of the parameters $p_1, \ldots, p_i$ can be discarded, because the corresponding sensors' observations cannot influence sensor $i$'s contribution to the team utility. In what follows, we will therefore use the notation $\mathbf{p}_i$ to indicate the vector of parameters to function $U_i$. This vector $\mathbf{p}_i$ contains only those decision variables on which sensor $i$'s utility depends. Since a sensor's contribution necessarily depends on the observations it collects itself, $p_i \in \mathbf{p}_i$ will always hold.

#### 5.1.2.3 Applying the Max-Sum Algorithm

As mentioned before, the fact that a sensor's utility depends only on a subset of other sensors' decision variables (as a result of Theorem 5.2 and Equation 5.1), and that the team utility is the sum of each sensor's utility (Equation 5.4), allows us to apply the max-sum algorithm to solve Equation 5.4 in a distributed fashion. The attractive properties of the max-sum discussed in Section 2.5.3.1 make this algorithm an ideal candidate, since it scales well with the number of sensors and exploits the fact that the interdependencies between sensors are sparse.

Now, Equation 5.4 is easily translated into a factor graph, on which the max-sum algorithm can be directly applied. In this factor graph, the utility functions $U_m(\mathbf{p}_m)$ are represented by function-nodes, while the decision variables $p_j$ are represented by variable nodes; an edge exists between a function $U_m(\mathbf{p}_m)$, and a variable $p_n$ if $p_n \in \mathbf{p}_m$.

The following example shows how Equation 5.4 is encoded as a factor graph for the coordination problem in Example 5.1.

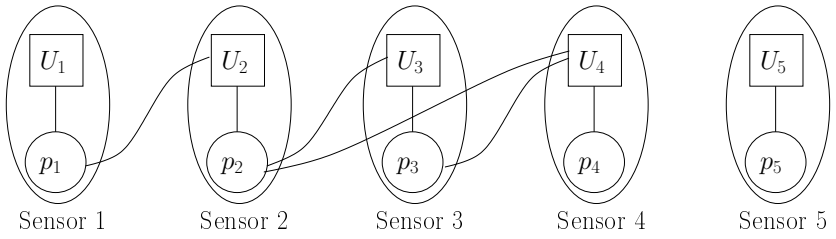



FIGURE 5.3: Factor graph encoding the coordination problem from Example 5.1.

**Algorithm 3** Algorithm for computing pruning message from function $U_m$ to variable $p_n$

1: $\overline{U_m}(p_n) \leq \min_{\mathbf{p_m} \setminus p_n} U_m(\mathbf{p}_m)$
2: $\underline{U_m}(p_n) \geq \max_{\mathbf{p_m} \setminus p_n} U_m(\mathbf{p}_m)$
3: send $\langle \overline{U_m}(p_n), \underline{U_m}(p_n) \rangle$ to $p_n$

**Example 5.2.** *Figure 5.3 shows an example factor graph that encodes Equation 5.4 for the coordination problem of Example 5.1. In this example, the utility of sensor 1 depends solely on its own action, so* $\mathbf{p}_1 = \{p_1\}$*; the the utility of sensor 2 depends on its own action, and that of sensor 1, so* $\mathbf{p}_2 = \{p_1, p_2\}$*. Similarly,* $\mathbf{p}_3 = \{p_2, p_3\}$*,* $\mathbf{p}_4 = \{p_2, p_3, p_4\}$*, and* $\mathbf{p}_5 = \{p_5\}$*.*

#### 5.1.2.4 Speeding up Message Computation

Unfortunately, the straightforward application of max-sum to solve Equation 5.4 is not practical, because the computation of the messages from function to variable (Equation 2.19) is a major bottleneck. A naïve way of computing these messages for a given variable $p_n$ is to enumerate all joint moves (i.e. the domain of $\mathbf{p}_m$), and evaluate $U_m$ for each of these moves. Since the size of this joint action space grows exponentially with both the number of sensors, and the number of available moves for each sensor, the amount of computation quickly becomes prohibitive. This is especially true when evaluating $U_m$ is costly, as is the case in the mobile sensors domain.[4] Therefore, we introduce two generic pruning algorithms to reduce the size of the joint action space that needs to be considered. These algorithms are then applied to our mobile sensor domain, but they can just as easily be applied within other settings.

**The Action Pruning Algorithm** The first algorithm attempts to reduce the number of moves each sensor needs to consider *before* running the max-sum algorithm. This algorithm prunes the dominated actions that can never maximise Equation 5.4, regardless of the actions of other sensors. More formally, a move $a' \in A_n$ is dominated if there exists

[4]Specifically, determining the value of a sample involves the inversion of a potentially very large matrix $K(\mathbf{X}, \mathbf{X})$ (see Equation 2.2).

**Algorithm 4** Algorithm for computing pruning messages from variable $p_n$ to all functions $U_m \in adj(p_n)$

1: **if** a new message has been received from all $U_m \in adj(p_n)$ **then**
2: $\quad \perp(p_n) = \sum_{U_m \in adj(p_n)} \underline{U_m}(p_n)$
3: $\quad \top(p_n) = \sum_{U_m \in adj(p_n)} \overline{U_m}(p_n)$
4: $\quad$ **while** $\exists a \in A_n : \top(a) < \max_{p_n} \perp(p_n)$ **do**
5: $\quad\quad A_n \leftarrow A_n \setminus \{a\}$
6: $\quad$ **end while**
7: $\quad$ send updated domain $A_n$ to each $U_m \in adj(p_n)$
8: **end if**

a move $a^*$ such that:

$$\forall \mathbf{a}_{-n} \sum_{U_m \in adj(p_n)} U_m(a', \mathbf{a}_{-n}) \leq \sum_{U_m \in adj(p_n)} U_m(a^*, \mathbf{a}_{-n}) \tag{5.5}$$

Just as with the max-sum algorithm itself, this algorithm is implemented by message passing, and operates directly on the variable and function nodes of the factor graph, making it fully decentralised:

- **From function to variable**: Function $U_m$ sends a message to $p_n$, containing the minimum and maximum values of $U_m$ with respect to $p_n = a_n$, for all $a_n \in A_n$. (see Algorithm 3).
- **From variable to function**: Variable $p_n$ sums the minimum and maximum values from each of its adjacent functions, and prunes dominated actions. It then informs neighbouring functions of its updated domain (see Algorithm 4).

Using this distributed algorithm, functions continually refine the bounds on the utility for a given state of a variable, which potentially causes more actions to be pruned. Therefore, it is possible that action pruning starts with a single move at a single sensor, and subsequently propagates through the entire factor graph. This algorithm terminates once the messages exchanged between the functions and variables converge. That is, when all messages along all edges in the factor graph equal the previously received messages. Also note that termination is guaranteed because of the fact that every variable has a finite number of states: during each iteration either at least one variable state is pruned or the algorithm has converged. To see why this is true, note that for the bounds on $U_m$ for a certain action $a$ to change, at least one variable state needs to get pruned. Otherwise, the messages sent from variables to functions will be identical, and all variables receive the same message twice, which results in the termination of the algorithm.

**Algorithm 5** Greedy algorithm for approximating the lower bound of $U_m(p_n)$

1: $\mathbf{p'}_m = \mathbf{p}_m$
2: $U'_m = U_m$
3: **for all** $p_i \in \mathbf{p'}_m \setminus \{p_n\}$ **do**
4: $\quad U'_m(\mathbf{p'}_m \setminus \{p_i\}) = \min_{p_i} U'_m(\mathbf{p'}_m)$
5: $\quad \mathbf{p'}_m = \mathbf{p'}_m \setminus \{p_i\}$
6: **end for**
7: **return** $U'_m(\mathbf{p'}_m)$ $\{= U'_m(p_n) = \underline{U_m}(p_n)\}$

Given the highly non-linear relations expressed in Equation 2.2, on which the sensors' utility functions $U_m$ are based, it is very difficult to calculate these bounds exactly, without exhaustively searching the domain of $\mathbf{p}_m$ for utility function $U_m$. Needless to say, this would defeat the purpose of this pruning technique. Nonetheless, experimentation showed that by computing these bounds in a greedy fashion, a very good approximation is obtained. Thus, the lower bound $\underline{U_m}(a_n)$ on a move $a_n$ is approximated by selecting the neighbouring sensors one at a time, and finding the move that reduces the utility of sensor $m$'s move the *most*. This idea is formalised in Algorithm 5; in lines 3—6 the algorithm iterates through the variables, and in line 4 the current variable is "minimised out". This process continues until only $p_n$ is left. The resulting function $U'_m(\mathbf{p'}_m)$ equals the desired bounds $\underline{U_m}(p_n)$, which encodes the lower bounds of all moves in the domain of $p_n$. Now, in a similar vein, the upper bound $\overline{U_m}(a_n)$ on a single move $a_n$ is obtained by selecting those moves of other sensors that reduce the utility the *least*. Thus, by substituting max for min in line 4 of Algorithm 5, we obtain an algorithm for approximating $\overline{U_m}(p_n)$

**The Joint Action Pruning Algorithm** Whereas the first algorithm runs as a pre-processing phase to max-sum, the second algorithm is geared towards speeding up the computation of the messages from function to variable (Equation 2.19), *while* max-sum is running. A naïve way of computing this message to a single variable $p_i$ is to determine the maximum utility for each of sensor $i$'s actions by exhaustively enumerating the joint domain of the variables in $\mathbf{p}_m \setminus \{p_i\}$ (i.e. the Cartesian product of the domains of these variables), and evaluating the expression between brackets in Equation 2.19, which we denote by:

$$\tilde{R}_{U_m \to p_n}(\mathbf{p}_m) = U_m(\mathbf{p}_m) + \sum_{\substack{p_{n'} \in adj(U_m) \\ p_{n'} \neq p_n}} Q_{p_{n'} \to U_m}(p_{n'}) \tag{5.6}$$

However, instead of just considering joint moves, we now allow some actions to be undetermined, and thus, consider *partial* joint moves, denoted by $\hat{\mathbf{a}}$. By doing so, we can create a search tree on which we can employ branch and bound to significantly reduce the size of the domain that needs to be searched. In more detail, to compute $R_{U_m \to p_n}(a_n^i)$

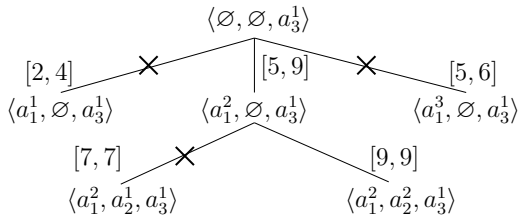



FIGURE 5.4: Search-tree for computing $R_{U_m \to x_3}(a_3^1)$ showing lower and upper bounds on the maximum value in the subtree.

(a single element of the message from $U_m$ to variable $p_n$) for a single action $a_n^i \in A_i$ in the domain of $p_n$, we create a search tree $\mathcal{T}(a_n^i)$ as follows:

- The root of $\mathcal{T}(a_n^i)$ is a partial joint move $\hat{\mathbf{a}}_r = \langle \varnothing, \ldots, \varnothing, a_n^i, \varnothing, \ldots, \varnothing \rangle$, which indicates that $a_n^i$ is assigned to $p_n$, and the remaining variables are unassigned (denoted by $\varnothing$).
- The children of a vertex $\langle a_1^{(1)}, \ldots, a_k^{(k)}, \varnothing, \ldots, \varnothing, a_n^i, \varnothing, \ldots, \varnothing \rangle$ are obtained by setting the first unassigned variable $p_{k+1}$ to each of its $|A_{k+1}|$ moves.
- The leafs of the tree represent a (fully determined) joint move $\mathbf{a}_m$ (i.e. $\forall i : p_i \neq \varnothing$). In the tree, only leafs are assigned a value, which is equal to $\tilde{R}_{U_m \to p_n}(\mathbf{a}_m)$.

The maximum value found in $\mathcal{T}(a_n^i)$ is the desired value. Now, in order to use branch and bound to find this value, we need to put bounds on the maximum value found in a subtree of $\mathcal{T}(a_n^i)$. These bounds depend on $U_m$ and the received messages $Q$. Now, in many cases we can put bounds on the maximum of the former, that is obtained by further completing a partial joint move $\hat{\mathbf{a}}'$ in a subtree of $\mathcal{T}(a_n^i)$. The bounds on $U_m$, combined with the minimum and maximum values of $Q$ for $\hat{\mathbf{a}}'$ (again, by further completing the partial joint move), gives us the desired bounds.

In more detail, Figure 5.4 shows an example of a partially expanded search tree for computing a single element $R_{U_m \to x_3}(a_3^1)$ of a message from function $U_m$ to variable $p_3$. Given the lower and upper bounds on the maximum (denoted between brackets) subtree $\langle a_1^1, \varnothing, a_3^1 \rangle$ can be pruned immediately after expanding the root. Similarly, subtree $\langle a_1^3, \varnothing, a_3^1 \rangle$ is pruned after expanding leaf $\langle a_1^2, a_2^2, a_3^1 \rangle$, which has the desired maximum value.

Since the utility functions $U_m$ are domain dependent in the context of the max-sum algorithm, there is no general way of computing these bounds. However, in most domains, such as the mobile sensor domain, a partial joint move has a meaningful interpretation that can lead to an intuitive way of computing the bounds on $U_m$ in any subtree of $\mathcal{T}$: a partial joint move $\hat{\mathbf{a}}$ represents a situation in which only a subset of the sensors have determined their move (Figure 5.5(a)). With this interpretation, we can obtain bounds

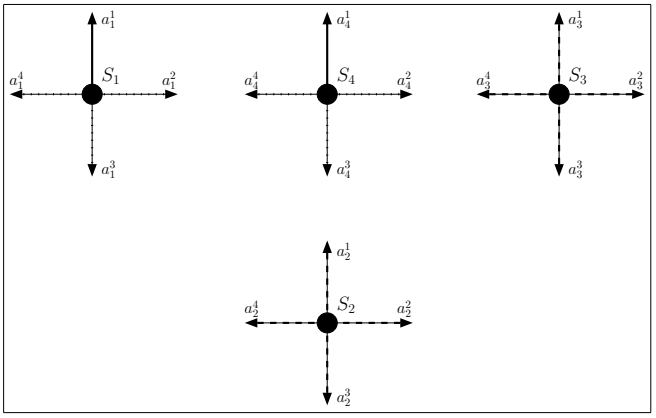
as follows. The upper bound on this value is obtained by disregarding the sensors that have not determined their action (i.e. sensors $i$ for which $p_i = \varnothing$). Since the act of collecting a sample always reduces the value of other samples, disregarding the samples of these 'undecided' sensors will give an upper bound on the maximum (Figure 5.5(b)). To obtain a lower bound on the maximum, we use the locality property of the utility functions, which tells us that the interdependency between values of samples weakens as their distance increases. So, in order to obtain a lower bounds on the maximum, we move the undecided sensors away from sensor $i$'s destination (Figure 5.5(c)).

### 5.1.3 Ensuring Network Connectivity

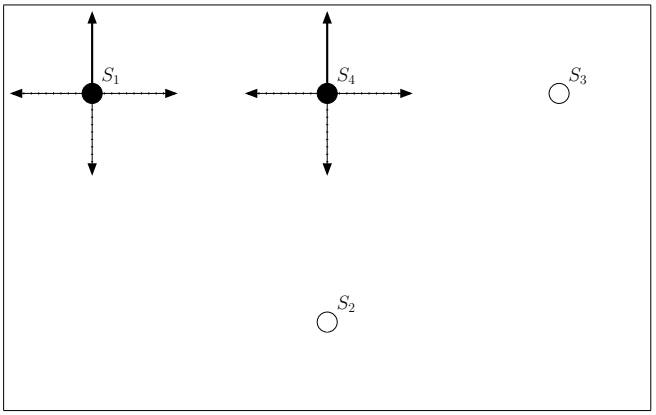
In many situations, it is also important that the sensors maintain network connectivity in order to transmit their measurements to a base station. More importantly in the context of our algorithm, sensors need to be able to communicate in order coordinate their actions. Not surprisingly, we can use the algorithm to accomplish this, by penalising disconnection from the network in the utility function $U_i$. To this end, we assume that every sensor maintains a routing table that specifies which sensors can be reached through each immediate neighbour. Thus, a move is allowed if all sensors are still be reachable through the remaining links. Otherwise, the sensor risks disconnection from the network, in which case a large penalty is added to its utility function $U_i$.

It is important to note, however, that although maintaining network connectivity is incentivised, the sensors are not *guaranteed* to remain connected. This is due to the fact that max-sum is not guaranteed to converge in cyclic graphs (see Section 2.5.3.1), and solutions can be arbitrarily bad when it does not. So, the solution computed by max-sum can disconnect the network, even though this results in a strong negative utility. We intend to address this problem in future work. Currently, we are investigating the use of a bounded version of max-sum (Farinelli et al., 2009) that operates on a non-cyclic subgraph of the factor graph. In so doing, it guarantees convergence and gives bounds on the computed solution, at the cost of no longer being able to compute the optimal solution. However, by utilising this extension to the max-sum algorithm, it might be possible to guarantee network connectivity by sacrificing a small amount of information value.

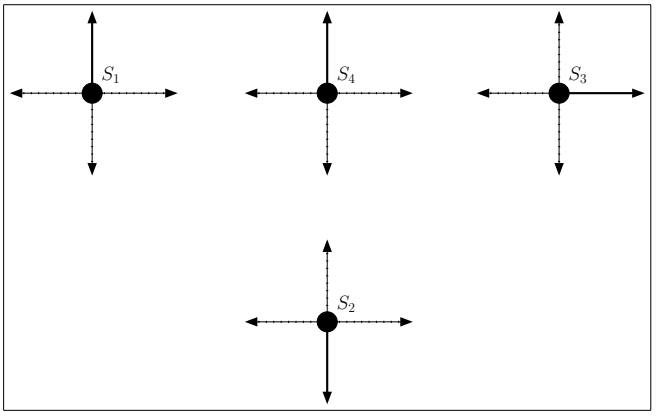
Now, more formally, let $\mathcal{R}_i^{\mathcal{C}} : S \rightarrow 2^S$ be the routing table for sensor $i$ in the communication network represented by graph $\mathcal{C}$, that maps a sensor to a set of sensors can be reached through it. If $j$ is not a neighbour of $i$ (i.e. no direct communication link exists), then $\mathcal{R}_i^{\mathcal{C}}(j) = \emptyset$. Also, we define $\mathcal{R}_i^{\mathcal{C}}(i) = \{i\}$. Given this routing table, a sensor can detect whether the communications network is connected by checking if $\bigcup_{j=1}^{M} \mathcal{R}_i^{\mathcal{C}}(j) = S$. When clear from the context, we will omit the superscript referring to the communication graph $\mathcal{C}$.




(a) A partial joint move $\langle a_1^1, \varnothing, \varnothing, a_4^1, \rangle$, in which sensors $S_1$ and $S_4$ move upwards, while sensors $S_2$ and $S_3$ have not determined their moves yet. (Arrows indicate chosen moves; dotted arrows represent moves that have not been chosen; dashed arrows are moves that can still be chosen.)




(b) For the joint move in (a), an upper bound is computed by ignoring (the moves of) sensors $S_2$ and $S_3$. In so doing, the entropy of the observations $S_4$ collects is not decreased by the observations of $S_2$ and $S_3$. As a result, this scenario puts an upper bound on the maximum obtainable utility of $S_4$.




(c) A lower bound on the maximum utility of sensor $S_4$ is obtained by moving sensors $S_2$ and $S_3$ *away* from the destination of $S_4$, thereby minimising the dependence between the observations of $S_2$ and $S_3$ and the observations $S_4$ collects.

FIGURE 5.5: Computing lower and upper bounds on the utility of a partial joint move.

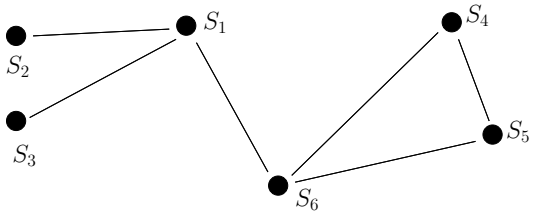



FIGURE 5.6: A communications network between sensors.

**Example 5.3** (Network Connectivity)**.** *Figure 5.6 shows a communications network among 6 sensors. The routing table for sensor 6 in this scenario is:* $\mathcal{R}_6(1) = \{1, 2, 3\}$, $\mathcal{R}_6(2) = \emptyset$ , $\mathcal{R}_6(3) = \emptyset$, $\mathcal{R}_6(4) = \{4, 5\}$, $\mathcal{R}_6(5) = \{4, 5\}$, $\mathcal{R}_6(6) = \{6\}$. *Now, since* $\{1, 2, 3\} \cup \{4, 5\} \cup \{6\} = S$, *the network is connected.*

Now, given the routing function $\mathcal{R}$, a sensor is able to determine if a move will disconnect it from the network. In particular, if a move results in the disconnection of one of its neighbours $j$, the network is still connected if $\bigcup_{i=1,i\neq j}^{M} \mathcal{R}(i) = S$. If, however, this condition does not hold, the sensor risks disconnecting itself from (a part of) the communications network.

**Example 5.4** (Network Connectivity, Continued)**.** *Suppose sensor 6 repositions itself, causing connection loss with sensor 4. Now,* $\bigcup_{i=1,i\neq 4}^{6} \mathcal{R}_6(i) = \{1, 2, 3, 4, 5, 6\} = S$ *so sensor 6 is still connected to all other sensors. Thus, any move by sensor 6 that disconnects it from sensor 4* only*, does not disconnect the network. If, however, it loses connection with sensor 1,* $\bigcup_{i=1,i\neq 1}^{M} \mathcal{R}_6(i) = \{4, 5, 6\} \neq S$, *and sensor 6 is no longer connected to sensors* 1, 2, *and* 3.

In more detail, let $C(p_1, \ldots, p_M)$ be the communication network after the sensors have moved according to their decision variables $p_1, \ldots, p_M$. We can now define a Network Connection Penalty that penalises moves that disconnect the communication network:

**Definition 5.4.** (Network Disconnection Penalty)

$$P_i^{Conn}(p_1, \ldots, p_M) = \begin{cases} 0 & \text{if } \bigcup_{j=1}^{M} \mathcal{R}_i^{C(p_1,\ldots,p_M)}(j) = S \\ -\infty & \text{otherwise} \end{cases}$$

Of course, since only neighbours influence the result of $\bigcup_{j=1}^{M} \mathcal{R}_i(j)$, only the variables $p_j$ of the sensors that are adjacent in the communications network need to be included in the set of parameters to $P_i^{Conn}$.

The penalty function is used by augmenting it to the original utility functions of the sensors.

**Algorithm 6** A decentralised algorithm for determining whether a communication network is fault tolerant.

**Require:** $\mathcal{R}_i^{\mathcal{C}}$: routing table for sensor $i$

**Ensure:** Returns **true** iff failure of sensor $i$ does not disconnect the communications network

*Initialisation:*

1: **for all** $j \in S \setminus \{i\}$ **do**
2: $\quad \mathbf{C} \leftarrow \mathbf{C} \cup \{\mathcal{R}_i(j) \cup \{j\}\}$
3: **end for**

*Closure:*

4: **while** $\exists C_1, C_2 \in \mathbf{C} : C_1 \cap C_2 \neq \emptyset$ **do**
5: $\quad \mathbf{C} \leftarrow \mathbf{C} \setminus \{C_1, C_2\}$
6: $\quad \mathbf{C} \leftarrow \mathbf{C} \cup \{C_1 \cup C_2\}$
7: **end while**
8: **return** **true** if $|\mathbf{C}| = 1$, **false** otherwise

**Definition 5.5** (Augmented Sensor Utility)**.**

$$\tilde{U}_i(\mathbf{p}_i) = U_i(\mathbf{p}_i) + P_i^{Conn}(\mathbf{p}_i)$$

In conjunction with max-sum, the augmented utility function incentivises the sensors to avoid moves that result in the disconnection of a sensor, since the team utility (see Equation 5.4) of such a move is $-\infty$.

More sophisticated forms of connectivity can also be incentivised with the same technique. In particular, *fault tolerance* is one of these. A communications network is said to be fault tolerant if it has at least two alternative paths between any pair of sensors, making it more robust against sensor failures. This coincides with the concept of 2-connected graphs, as established by Whitney's Theorem (Gross and Yellen, 1999). In more detail, 2-Connectivity implies that at least two sensors need to fail simultaneously before the network becomes disconnected (or is reduced to a single sensor). Clearly, the communications network from Example 5.3 is not 2-connected; the failure of sensor 6 breaks the network into two components: $\{1, 2, 3\}$ and $\{4, 5\}$

By inspecting its routing table, a sensor can easily determine if it satisfies the necessary condition for 2-connectivity: its failure should leave the communications network connected. In order to do this, Algorithm 6 is used to count the number of components the network is divided into should the sensor fail.

Now, the following penalty function can be used to augment the utility function of the sensor in order to enforce fault tolerance in a decentralised fashion.

**Definition 5.6** (2-Connectedness Violation Penalty)**.**

$$P_i^{2\text{-}Conn}(p_1, \ldots, p_M) = \begin{cases} 0 & \text{if } isFaultTolerant(i, C(p_1, \ldots, p_M)) \\ -\infty & \text{otherwise} \end{cases} \tag{5.7}$$

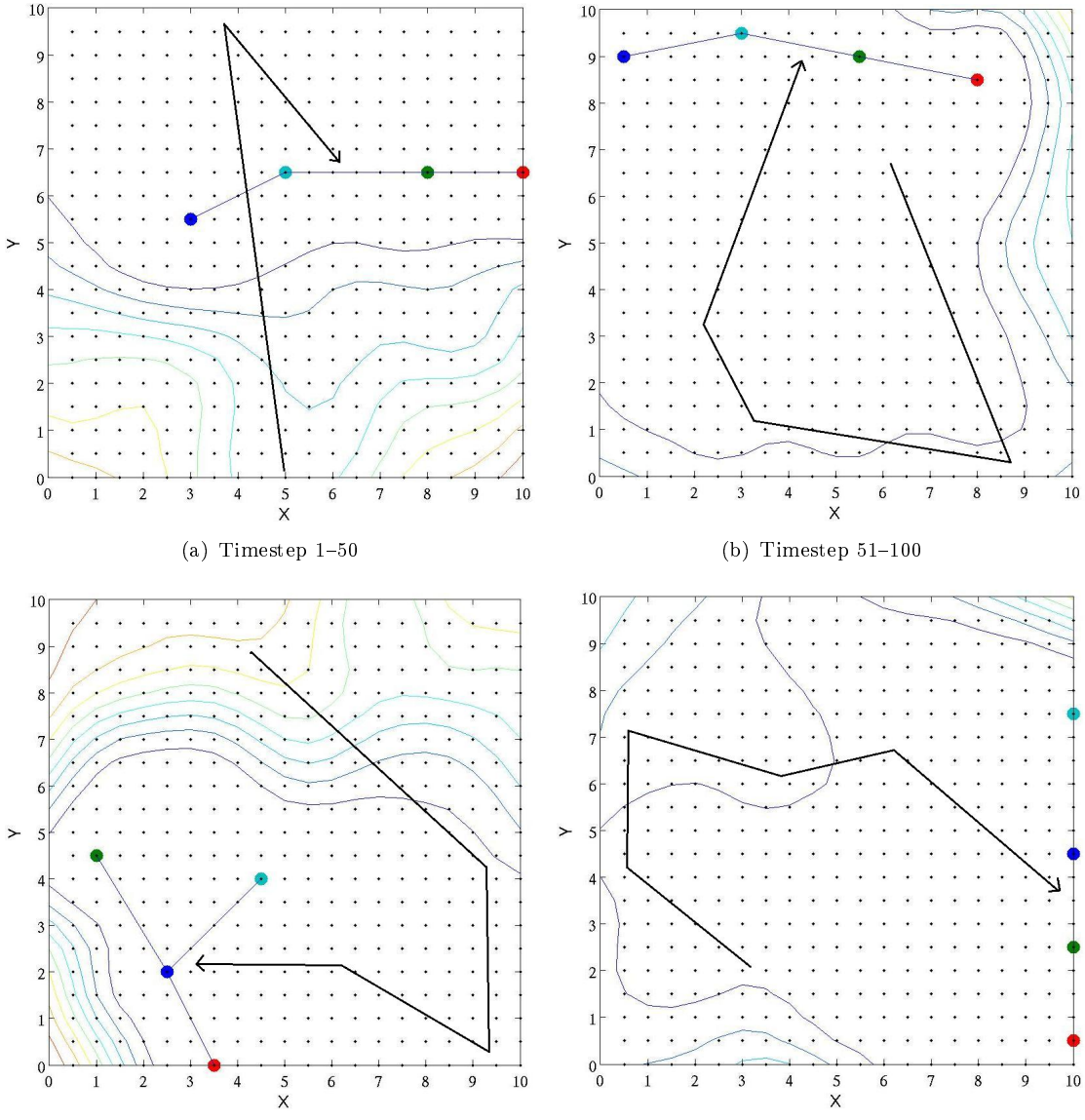
In general, the technique of augmenting the utility function with a penalty function is applicable to enforce various constraints on the sensors' collective movement. The only requirement on such a constraint is that its violation should be locally detectable. That is, one or more sensors should be able to determine if the constraint is satisfied based on information received from direct neighbours or from its own observations. If this is the case, a penalty function can be effectively used to filter out solutions that do not satisfy the constraint. Other examples, besides the ones given above, include the assignment of roles to sensors (i.e. some sensors function as communication relays, others as explorers, etc.), and to ensure certain events or locations are monitored by sensors.

## 5.2 An Example Scenario

In this section we describe an example scenario that illustrates the operation of the algorithm. In order to do so, we used the same simulator that was used to empirically evaluate the un-negotiated algorithm presented in the previous Chapter. Now, in this scenario, a team of four sensors are monitoring an environmental phenomenon in a rectangular environment. The sensors coordinate their movements to maximise information gain, while at the same time attempting to maintain network connection using the techniques described in the previous section. Figure 5.7 shows four snapshots of this simulation, taken at timesteps 50, 100, 150 and 200. Note how the team makes a sweeping motion through its environment, in order to maintain low uncertainty throughout the environment, and thus reducing the predictive variance with which the environmental phenomenon can be predicted. It is also worth noting that the team maintains connectivity throughout the simulation, and that the structure of the communication network changes over time. A video of this simulation that offers a more dynamic view of the motion of the sensors can be found at `http://users.ecs.soton.ac.uk/rs06r/videos/`.

## 5.3 Empirical Evaluation

In this section we empirically evaluate the algorithm developed in this chapter. First, we explain the experimental setup used to test the algorithm; we detail the simulated environment in which the sensors operate, and the benchmark strategies against which the algorithm was compared. Next, we present and analyse the results of these experiments.




(a) Timestep 1–50

(b) Timestep 51–100

(c) Timestep 101–150

(d) Timestep 151–200

FIGURE 5.7: Snapshots at timesteps 50, 100, 150 and 200 of a team of four sensors in coordinated negotiation mode. The grid points represent the locations the sensors can move between, and the arrows indicate the approximate path of the team in the 50 timesteps between two subsequent snapshots. The lines between the sensors indicate the communication network between them.

### 5.3.1 Experimental Setup

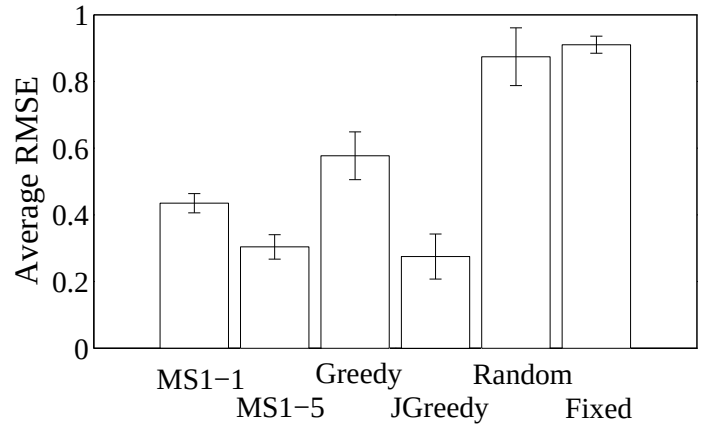
To empirically evaluate the algorithm, we simulated five sensors on a lattice graph measuring 26 by 26 vertices. The data was generated by a GP with a squared exponential covariance function (see Equation 2.4) with a spatial length-scale of 10 and a temporal length-scale of 150. This means that the spatial phenomenon has a strong correlation along the temporal dimension, and therefore changes slowly over time. At every $m$ time steps, the sensors plan their motion for the next $l$ time steps ($l \geq m$). In what follows, this strategy is referred to as MS$m$-$l$. Now, instead of considering all possible paths of length $l$ from a sensor's current position, which would result in a very high computational overhead, the action space is limited to the locations in $G$ that can be reached in $l$ time steps in 8 different directions, corresponding to the major directions on the compass rose. In the first experiment, we benchmarked MS1-1 and MS1-5 against four strategies:

- **Random:** Randomly moving sensors.
- **Greedy:** Sensors that utilise the un-negotiated coordination algorithm presented in the previous chapter.
- **J(umping) Greedy:** The same as Greedy, except that these sensors can instantaneously jump to any location.
- **Fixed:** Fixed sensors that are placed using the algorithm proposed in Guestrin et al. (2005).

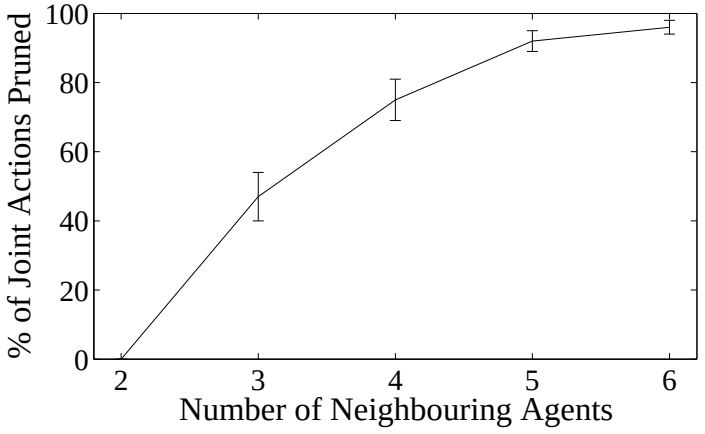
### 5.3.2 Results

The averaged root mean squared error (RMSE) for 100 time steps is plotted in Figure 5.8(a). From this, it is clear that both MS strategies outperform the Greedy, Random, and Fixed strategies. Furthermore, the prediction accuracy of MS1-5 is comparable to that of JGreedy, whose movement is not restricted by graph $G$. Moreover, it shows that increasing the length of the considered paths from 1 to 5, reduces the RMSE by approximately 30%.

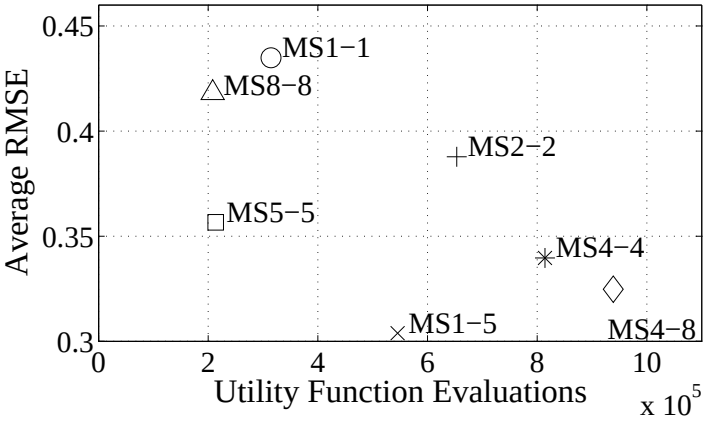
In the second set of experiments, we analysed the speed-up achieved by applying the two pruning techniques described in Section 5.1.2.4. To this end, Figure 5.8(b) shows the percentage of joint actions pruned plotted against the number of neighbouring sensors. With 5 neighbours, the two pruning techniques combined prune around 92% of the joint moves. With such a number of neighbouring sensors, the sensors are strongly clustered, which occurs rarely in a large environment. However, should this happen, the utility function needs to be evaluated for only 8% of roughly $8^5$ joint actions, thus greatly improving the algorithm's efficiency.




(a)




(b)




(c)

FIGURE 5.8: The Experimental Results. Errorbars indicate the standard error in the mean.

In the third experiment, we performed a cost/benefit analysis of various MS$m$-$l$ strategies. More specifically, we examined the effect of varying $m$ and $l$ on both the number of utility function evaluations, and the resulting RMSE. In more detail, Figure 5.8(c) shows the results. The results of MS1-1, MS2-2, MS4-4, MS5-5, and MS8-8 show an interesting pattern. Up to and including $m = l = 4$, both the number of function evaluations and the average RMSE decrease. This is due to the fact that planning longer paths is more expensive, but results in lower RMSE. However, for $m, l > 4$, the action space becomes too coarse (since only 8 directions are considered) to maintain a low RMSE. At the same time, the number of times the sensors coordinate reduces significantly, resulting in a lower number of function evaluations. Finally, MS1-5 and MS4-8 provide a compromise; they compute longer paths, but coordinate more frequently. This leads to more computation compared to MS5-5 and MS8-8, but results in significantly lower RMSE, because sensors are able to 'reconsider' their paths.

## 5.4 Summary

In this chapter, we presented an algorithm for maximising the joint utility of the sensors as a team. This algorithm operates in negotiated coordination mode, in which sensors negotiate about their moves prior to executing them. We showed that this leads to significant improvement over the greedy un-negotiated algorithm presented in the previous chapter. Moreover, we have shown that the max-sum algorithm can be used to facilitate this negotiation.

More specifically, the algorithm presented in this chapter results in several improvements in terms of some the requirements stated in the introduction:

**Accuracy:** The empirical results from 5.3 show that the algorithm outperforms the greedy un-negotiated algorithm in terms of prediction error, and is therefore more accurate.

**Relevance:** The improvements made in this chapter have not addressed the requirement of relevance.

**Adaptiveness:** Since the algorithm developed in this chapter makes it possible to plan multiple timestep ahead, instead of being limited to single steps, the team of sensors is capable of adapting better to its environment.

**Robustness:** The max-sum algorithm—on which the coordination algorithm presented in this chapter is based—is robust to message loss and the failure of sensors. Moreover, we have extended the algorithm from previous chapter with the possibility of incentivising network connectivity between the sensors. As a result, the probability of successful communication between sensors and a base station has increased, thereby increasing the robustness of the information stream.

**Autonomy:** The algorithm maintains the *autonomy* of the sensors.

**Scalability:** The use of the max-sum algorithm scales well with the size of the team, since computation and communication scale with the number of neighbours, not with the size of the team. This algorithm is therefore scalable to the same extent as the un-negotiated algorithm.

**Modularity:** The improvements made in this chapter have not addressed this requirement.

Thus far, we have considered algorithms that impose a discrete structure on the layout of the environment, and as a result, a discrete action model for the sensors. In this model, sensors have a (finite) discrete number of ways to reposition themselves (as defined by movement graph $\mathcal{G}$). In reality, however, this motion model might be too restrictive, and could result in sub-optimal solutions (in terms of *accuracy*). To prevent this, a more natural motion model in which the sensors set their heading and speed is more appropriate. This model does more justice to the freedom of motion they possess in reality. To this end, in the next chapter, we develop an extension to the max-sum algorithm that allows for continuously valued actions.

# Chapter 6

# Negotiated Coordination with Continuously Valued Actions

Thus far, we have considered mobile sensors whose motion is restricted by a discrete structure; a graph $\mathcal{G}$ that encodes the environment. Naturally, such a structure may be used as an abstract representation of the sensors' environment, and—as we have shown in previous chapters—discrete algorithms can be successfully applied it. However, there are many scenarios in which the motion of the sensors is governed by continuous control parameters, and thus, a discrete structure is lacking. A relevant example of this is the deployment of UAVs: the space in which they move is not characterised by a discrete layout, and their motion is controlled by adjusting pitch, roll, yaw, and trust, all of which are continuous control parameters. Now, in these cases, actions that are computed by a negotiation algorithm that operates in a discrete action space need to be post-processed in order provide valid input to the interface of the sensor's hardware. In other words, discrete signals need to be converted to analogue control inputs. However, a negotiation algorithm that directly operates in the native (continuous) action spaces of the underlying platform can avoid such a translation step, and select an optimal action from a continuous set. This can potentially lead to increased accuracy, since the range of actions that is considered equals the range of possible inputs, instead of only a finite set.

Moreover, in terms of the requirement of *accuracy* stated in the introduction, we note that whilst these continuous control parameters may be discretised in order to apply existing DCOP algorithms, such discretisation results in sub-optimal solutions even when complete DCOP algorithms are used (see Section 2.5.3.1). Furthermore, in terms of the *scalability* requirement, increasing the discretisation in order to improve the solution quality rapidly increases the computation and communication overhead of the coordination.

Thus, against this background, there is a clear need for a distributed constraint optimisation algorithm that can be applied within these applications to make coordinated decisions regarding continuous valued control parameters. It is this requirement that we address in this chapter,[1] and to this end, we present a novel distributed constraint optimisation algorithm for continuous control parameters that can be applied to any setting where the interaction between the devices can be described by piecewise linear utility functions. We choose this particular class of utility function because it can be manipulated using standard techniques from computational geometry, resulting in a computationally efficient algorithm, but at the same time it is a general class of function that can be used to approximate any arbitrary continuous function.

As in the previous chapter, the max-sum algorithm will be the starting point for developing this algorithm. Despite the attractive properties of the max-sum algorithm discussed in Section 2.5.3.1, and the fact that the GDL framework on which it is based is very general, it has never before been used in the context of distributed optimisation to solve problems involving variables with continuous domains. It is this specific challenge that we address in this chapter, and to do so requires that we redefine the fundamental operations of the max-sum algorithm (i.e. addition of two utility functions and marginal maximisation) for the continuous space in which they now operate.

After doing so, we will demonstrate and empirically evaluate this algorithm by applying it to the problem of coordinating the sense/sleep cycle of energy constrained sensors deployed within a network for wide-area surveillance. We chose to empirically evaluate the algorithm in the wide-area surveillance setting, rather than apply it to the mobile sensor domain, because the interactions between the mobile sensors are not piecewise linear. Instead, they are determined by the Gaussian process equations, which are determined by strongly non-linear covariance functions. In contrast, the interactions between the sensors in the wide-area surveillance setting can be exactly represented by piece wise linear functions. Moreover, this setting has been extensively used to evaluate the max-sum algorithm for discrete action spaces (Farinelli et al., 2008). Hence, empirical evaluation within this setting facilitates comparison of our approach with this algorithm.

Thus, in more in detail, we make the following contributions:

1. We derive a representation that uses simplexes to describe the utility interactions of continuous valued control parameters (or variables). We then derive exact algorithmic solutions for the fundamental operations required to apply the max-sum algorithm in the domain of continuous control parameters and piecewise linear utility functions; specifically, addition and marginal maximisation of general n-ary piecewise linear functions.

[1] This chapter is based on Stranders et al. (2009a).

2. We demonstrate the applicability of this algorithm by applying it to the problem of coordinating the sense/sleep cycles of energy constrained sensors such that the system-wide probability of detecting an event is maximised.

3. We empirically evaluate the performance of our continuous max-sum algorithm in this setting by comparing it against the conventional discrete version in which the continuous valued control parameter is artificially discretised. We show that the continuous max-sum algorithm is able to provide a more accurate solution (with up to a 10% increase in solution quality), while allowing a more compact representation of the utility functions. This, in turn, results in a lower coordination overhead in terms of message size (the continuous max-sum algorithm shows a reduction of up to half the total message size).

The remainder of this chapter is structured as follows: in Section 6.1 we formally describe the social welfare maximisation problem that we face. In Section 6.2 we present our representation of the utility as piecewise linear functions, and the way we perform the operations needed by the max-sum algorithm. In Section 6.3 we describe our test-bed and present our experimental results, before concluding in Section 6.4.

## 6.1 Problem Description

We now formally describe the decentralised coordination problem that we address in this report. In Section 6.3 we instantiate this general problem to the specific case of a sensor network composed of energy constrained sensors performing a wide-area surveillance task. We consider a set of $M$ sensors, each of which has a single[2] continuously valued control parameter $p_m$ whose domain is a closed and bounded interval in $\mathbb{R}$. Each sensor interacts directly with a set of other sensors, such that the utility of a sensor $m$, $U_m(\mathbf{p}_m)$, is dependent on the value of its own control parameter and that of those other sensors with which it interacts (defined by the vector $\mathbf{p}_m$). For example, in the specific wide area surveillance problem that we will detail in Section 6.3.1, the control parameter of each sensor represents the time at which it decides to activate its sensor, the interactions that arise through its sensor's sensing field overlapping with those of other nearby sensors, and the utility describes the probability that any sensor will detect an event. As stated previously, we focus on utility functions that can be represented as piecewise linear functions. Therefore, $U_m(\mathbf{p}_m)$ is a multivariate piecewise linear function for every sensor $m$.

As discussed in Section 2.5.3.1, we wish to find the value of each sensor's control parameter, $\mathbf{p}^*$, such that the sum of the individual sensors' utilities (or social welfare) is maximised:

[2]Note that our representation allows multiple control parameters per sensor, with possibly different domains, but for ease of exposition we present just the single case here.

$$\mathbf{p}^* = \arg\max_{\mathbf{p}} \sum_{i=1}^{M} U_i(\mathbf{p}_i) \tag{6.1}$$

As before, in order to enforce a truly decentralised solution, we assume that each sensor only has knowledge of, and can directly communicate with, the few neighbouring sensors that influence its own utility directly.

As mentioned earlier, the point of departure for developing a new algorithm that allows sensors to negotiate about continuously valued actions is the max-sum algorithm, which was detailed in Section 2.5.3.1. For convenience, we repeat the equations for computing the messages that are exchanged between functions and variables below:

- **From variable to function:**

$$Q_{p_n \to U_m}(p_n) = \sum_{\substack{U_{m'} \in adj(p_n) \\ U_{m'} \neq U_m}} R_{U_{m'} \to p_n}(p_n) \tag{6.2}$$

- **From function to variable:**

$$R_{U_m \to p_n}(p_n) = \max_{\mathbf{p_m} \setminus p_n} \left[ U_m(\mathbf{p}_m) + \sum_{\substack{p_{n'} \in adj(U_m) \\ p_{n'} \neq p_n}} Q_{p_{n'} \to U_m}(p_{n'}) \right] \tag{6.3}$$

Note these equations apply equally well to both continuously valued and discrete variables. However, the summation and marginal maximisation operations required in Equations 6.2 and 6.3 are much more readily implemented in the case of discrete variables. In the next section, we describe an efficient representation that uses computational geometry to describe the utility interactions over continuous valued variables, where these interactions can be represented by continuous piecewise linear functions. This then allows us to derive exact algorithmic solutions for these operations.

## 6.2 Max-Sum in Continuous Action Spaces

As described above, in order to apply the max-sum algorithm to cases in which the variables can take on continuous values, we need to be able to express each individual sensor's utility $U_i(\mathbf{p}_i)$ as a function of these continuous variables, and perform the fundamental operations of the max-sum algorithm on these functions. As mentioned earlier, we restrict our attention to cases in which this function $U_i(\mathbf{p}_i)$ is a multivariate continuous piecewise linear function (CPLF). We choose this particular class of utility

function because they represent a general class that can be used to approximate any arbitrary continuous function. Furthermore, under this restriction, the two aforementioned operations have an attractive geometric interpretation that makes it possible to define and manipulate them using standard techniques from computational geometry, hence allowing the continuous versions of the operations required by max-sum to be performed. Thus, more specifically, in this section we show how to:

1. Represent each sensors' utility function as a continuous piecewise linear function (CPLF).
2. Perform the summation of two continuous piecewise linear functions. This is in order to perform the necessary summation of the utility function, $U_j(\mathbf{p}_j)$, and the summation of incoming messages,

$$\sum_{\substack{p_{n'} \in adj(U_m) \\ p_{n'} \neq p_n}} Q_{p_{n'} \to U_m}(p_{n'})$$

   shown in Equation 6.3.
3. Calculate the marginal maximisation of a continuous piecewise linear function with respect to a single variable. This is in order to perform the necessary max operation in Equation 6.3.

We next present a formal definition of a CPLF, before going on to derive exact algorithmic solutions for computing the sum of two CPLFs, and the marginal maximisation of a CPLF with respect to a single variable. Finally, we connect the formalism developed here to the max-sum algorithm as described in the previous section.

### 6.2.1 Representing CPLFs with Simplexes

A continuous piecewise linear function is a function whose domain can be partitioned into a set of convex polytopes,[3] such that it is linear on each of these polytopes. For one variable, a CPLF is a function that can be represented with a finite number of line segments, while a CPLF of two variables can be represented by a finite number of two-dimensional polygons. An example of the latter is given in Figure 6.1. The domain of the function in this figure is partitioned into triangles (shown on the $(x_1, x_2)$ plane) on which the function is linear.

[3] A convex polytope is a multi-dimensional generalisation of the two-dimensional convex polygon. In $n$ dimensions, it is a convex hull of at least $n + 1$ points.

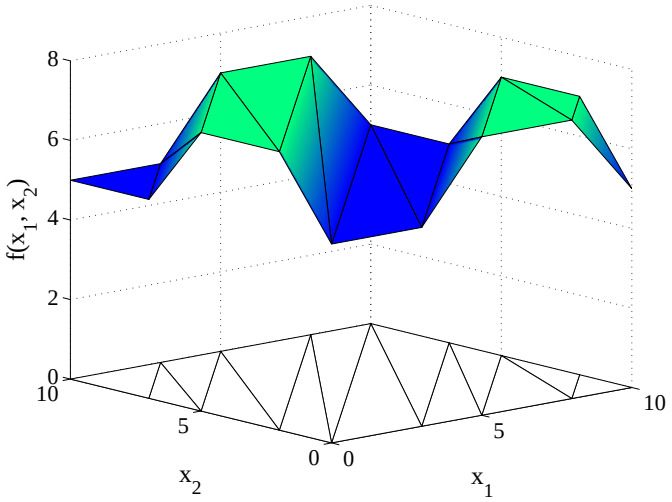



FIGURE 6.1: An example of a CPLF in two dimensions. This CPLF encodes the utility of two sensors $S_1$ and $S_2$ in the wide area surveillance scenario described in Section 6.3.1. Sensor $S_1$ can be active for $l_1 = 2$ out of every $L = 10$ time units, while $S_2$ can be active for $l_2 = 5$ out of every 10 time units.

In our formalism, we use $n$-dimensional simplexes, or $n$-simplexes, to partition the domain of an $n$-ary CPLF. The reason for this is that an $n$-simplex is the simplest $n$-dimensional polytope and it is therefore easy to manipulate. More specifically, an $n$-simplex is constructed by taking the convex hull of a set of $n+1$ affinely independent vertexes $\{\mathbf{p}_1, \ldots, \mathbf{p}_{n+1}\} \in \mathbb{R}^m$ $(m \geq n)$, and is denoted by $\Delta^n$. We will omit the superscript $n$ when its value is clear from the context. The set of points enclosed by a simplex is given by the following equation:

$$\Delta^n = \left\{ \mathbf{x} \in \mathbb{R}^m | \sum_{i=1}^{n} a_i \mathbf{p}_i = \mathbf{x}, \sum_i a_i = 1, \forall i : a_i \geq 0 \right\} \tag{6.4}$$

Now, an $n$-ary CPLF $f : \mathcal{D} \rightarrow \mathbb{R}$ is defined by a set of $n$-simplexes in $\mathbb{R}^{n+1}$: $\{\Delta_1, \ldots, \Delta_m\} \subset \mathbb{R}^{n+1}$. Here, domain $\mathcal{D}$ is the Cartesian product of the domains of its variables $(x_1, \ldots, x_n)$:[4] $\mathcal{D} = \mathcal{D}_{x_1} \times \cdots \times \mathcal{D}_{x_n}$. Since each $x_i$ is a scalar, $\mathcal{D}_{x_i}$ is a closed interval in $\mathbb{R}$. Consequently, $\mathcal{D}$ is an interval in $\mathbb{R}^n$, or an $n$-cube. From the definition of a CPLF, we require that the projection of the simplexes that make up $f$ is a partition $P_{\mathcal{D}}$ of $\mathcal{D}$. More formally,

$$P_{\mathcal{D}} = \left\{ \widetilde{\Delta}_i | 1 \leq i \leq m, \bigcup_i \widetilde{\Delta}_i = \mathcal{D}, \widetilde{\Delta}_i \bigcap \widetilde{\Delta}_j = \emptyset, 1 \leq i < j \leq m \right\} \tag{6.5}$$

where $\widetilde{\Delta}_i$ is the projection of $\Delta_i$ onto the $\mathbf{x}$ hyperplane.

[4] In what follows, $(x_1, \ldots, x_n)$ and $\mathbf{x}$ are used interchangeably.

Furthermore, in order to guarantee that $f$ is continuous, we must ensure that simplexes whose projection onto the $\mathbf{x}$ hyperplane share the same coordinates in $\mathbb{R}^n$, also share the same coordinates in $\mathbb{R}^{n+1}$.

Given this representation, we can now derive exact algorithmic solutions for computing the two fundamental operations required for implementing our continuous max-sum algorithm.

### 6.2.2 Summation of Two CPLFs

In order to perform the summation of two CPLFs $g$ and $h$ with identical domains, we need to compute the simplexes that make up function $f$ such that $\forall \mathbf{x} \in \mathcal{D} : f(\mathbf{x}) = g(\mathbf{x}) + h(\mathbf{x})$ holds. We denote the operator that adds two CPLFs as $\oplus$, which works in two steps. First, we need to compute a domain partition $P_f$ of $f$, such that $P_f$ contains a corner[5] at every corner in $g$ and $h$. Second, we need to compute the values of $f$ at each vertex $\mathbf{p}$ of the simplexes that partition $f$. The latter step is trivial; for each vertex $\mathbf{p}$, evaluate $g(\mathbf{p})$ and $h(\mathbf{p})$ and add them together. However, the former step is a little more involved, since computing the domain partition of $f$ involves overlaying or *merging* the partitions $P_g$ and $P_h$ in order to determine where the sum of $g$ and $h$ might have a corner. Algorithm 7 details how to perform this partition and it proceeds in two main steps:

1. Copy partition $P_g$ to the variable $P_f$ that contains the result while it is constructed (line 1).
2. Merge every simplex in $P_h$ with $P_f$ by overlaying it on $P_f$:
   (a) Add the vertexes of all simplexes in $P_h$ to $P_f$ (lines 2 to 6).
   (b) Add the edges of the simplexes in $P_h$ to $P_f$ (lines 7 to 13). These edges are the corners of $h$, and therefore need to be present in $P_f$.

As an example, Figures 6.2(c) shows the output of Algorithm 7 on the partitions shown in Figures 6.2(a) and 6.2(b). Note that the partition of $f$ shown in Figure 6.2(c) indeed has corners at every location where function $g$ and $h$ have corners.

To complete the specification of Algorithm 7, we need to define the split operator $\mathcal{S}$ that is essential in the merge process (lines 4 and 10). Specifically, the split operator $\mathcal{S}$ partitions a simplex $\Delta^n$ around a point $\mathbf{x}$: $\mathcal{S}(\Delta^n, \mathbf{x}) = \{\Delta^n_1, \dots \Delta^n_m\}$. Thus, each $\Delta^n_i \in \mathcal{S}(\Delta^n, \mathbf{x})$ is obtained by creating a simplex with vertexes $\{\mathbf{x}, \mathbf{p}_1, \dots, \mathbf{p}_{n+1}\} \setminus \{\mathbf{p}_i\}$. Depending on the location of $\mathbf{x}$ in $\Delta^n$, the split operator creates a different number of

[5] A corner is the location at which two simplexes meet at a factor graph is an an angle.

**Algorithm 7** An algorithm for merging two partitions

***Require:*** *Partitions* $P_g$ *and* $P_h$
***Ensure:*** *Partition* $P_f = P_g \oplus P_h$
1: $P_f \leftarrow P_g$
2: ***for all*** $\Delta \in P_h$ ***do***
3: ***for all*** $\mathbf{p} \in \{\mathbf{p}_1^\Delta, \ldots, \mathbf{p}_{n+1}^\Delta\}$ ***do***
4: $P_f \leftarrow \mathcal{S}(P_f, \mathbf{p})$
5: ***end for***
6: ***end for***
7: ***for all*** $\Delta_h \in P_h$ ***do***
8: ***for all*** $\Delta_f \in P_f$ ***do***
9: ***for all*** *intersections* $\mathbf{p}$ *of the edges of* $\Delta_h$ *with the* $(n-1)$*-faces of* $\Delta_f$ ***do***
10: $P_f \leftarrow \mathcal{S}(P_f, \mathbf{p})$
11: ***end for***
12: ***end for***
13: ***end for***
14: ***return*** $P_f$

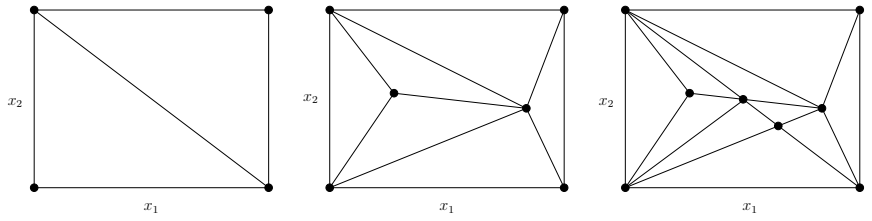



FIGURE 6.2: (a) Domain partition $P_g$ of function $g$. (b) Domain partition $P_h$ of function $h$. (c) Merged partition $P_f$ of function $f = g \oplus h$.

simplexes. In more detail, depending on the complexity of the face[6] of $\Delta^n$ on which $\mathbf{x}$ lies, $\mathcal{S}$ splits $\Delta^n$ into at least 1, and at most $n$ simplexes. Figures 6.3(b) and 6.3(c) show how the 2-simplex in 6.3(a) is split on points on a 2-face (body) and a 1-face (edge) respectively. Note that, in the latter case, the simplex is split in two, since vertexes $\{\mathbf{p}_1, \mathbf{p}_2, \mathbf{x}\}$ are not affinely independent, and therefore do not form a simplex. Splitting on a 0-face (or vertex) does not split the simplex, neither does splitting on a point outside the simplex. To avoid cluttering the notation in Algorithm 7, we denote the operation of splitting all simplexes in a partition $P$ on a point as $\mathcal{S}(P, \mathbf{x})$, which is shorthand for $\cup_{\Delta \in P} \mathcal{S}(\Delta, \mathbf{x})$.

### 6.2.3 Marginal Maximisation of a CPLF

Marginal maximisation is the second operator that is needed in max-sum. It takes as input a function $y = f(x_1, \ldots, x_n)$, and computes a single-dimensional CPLF $f(x_i) =$

[6]The faces of a simplex are simplexes that make up its boundaries. The *complexity* of a face of an $n$-simplex is its dimensionality, which ranges from 1 to $n$. A face of complexity $i$ is called an $i$-face. A 0-face is a vertex of the simplex, a 1-face is an edge, a 2-face is a triangle, etc. The $n$-face of the simplex is the simplex itself, which is also referred to as the *body*.

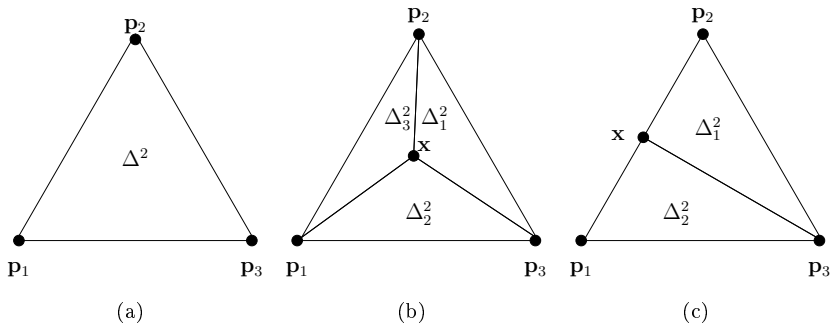



FIGURE 6.3: (a) A 2-simplex. (b) Splitting a 2-simplex on point $\mathbf{x}$ on a 2-face. (c) Splitting a 2-simplex on point $\mathbf{x}$ on a 1-face (edge)

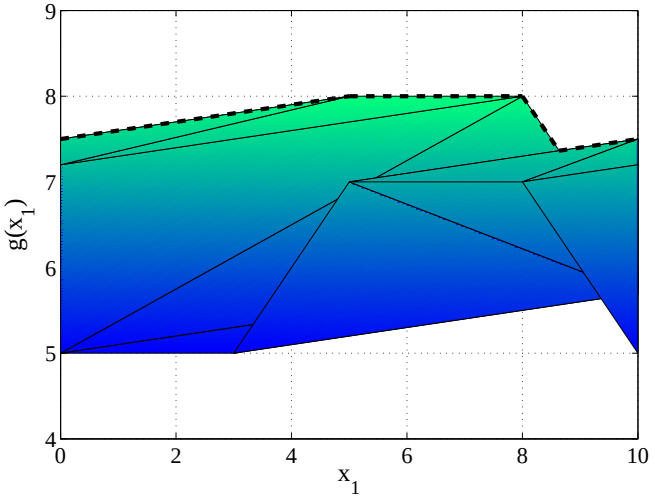



FIGURE 6.4: A CPLF $y = g(x_1, x_2)$ projected onto the $(x_1, y)$ plane. The dotted line indicates the upper envelope of these simplexes, and equals $g(x_1) = \max_{x_2} g(x_1, x_2)$.

$\max_{\mathbf{x}\setminus x_i} f(x_1, \ldots, x_n)$. The computation of the marginal maximisation of a CPLF proceeds in two steps:

1. Project all simplexes of $f$ onto the $(x_i, y)$ plane. An n-simplex $\Delta$ is projected by projecting each of its $m = \binom{n}{2}$ edges to obtain a set of line segments $S_\Delta = \{s_1, \ldots, s_m\}$.

2. Extract the *upper envelope* of the line segments in $S_\Delta$. The upper envelope is a function $\hat{U}_S$ of the set $S$ of all projected line segments of all simplexes that make up $f$ is then a function:

$$\hat{U}_S(x) = \max\{s(x) | s \in S \wedge x \in [s_s, s_e]\}$$

where $[s_s, s_e]$ is the closed interval on which line segment $s$ is defined. The upper envelope of a set of $n$ line segments can be computed in $O(n \log n)$ operations (Hershberger, 1989).

Figure 6.4 shows an example of this operation. In this figure the function $y = g(x_1, x_2)$ is projected onto the $(x_1, y)$ plane. This function $g(x_1, x_2)$ is the sum of the function from Figure 6.1 and the function $h(x_1, x_2) = 0.1x_2$ that slightly shears it along the $x_2$ axis.[7] The figure also shows the upper envelope of this projection that is the result of applying the marginal maximisation operator on this function $g$ with respect to variable $x_1$.

### 6.2.4 Instantiating the Continuous Max-Sum Algorithm

Now that we have defined the CPLF and the two required operations, we can instantiate the max-sum algorithm for continuous variables, by defining the processes through which messages between the variables and functions are computed:

- From variable to function (Equation 6.2). Since the messages $R_{U_k \to U_i}(x_i)$ are real-valued functions of a single continuous variable $x_i$, computing $Q_{U_i \to U_j}$ involves summing over single-dimensional CPLFs using the $\oplus$ operator of Section 6.2.2. The addition of scalar $a_{ij}$ is then a trivial operation.
- From function to variable (Equation 6.3). The computation of the message $R_{k \to i}(x_i)$ proceeds in two steps. First, the expression between the brackets is evaluated. The first term in this expression is the utility of sensor $j$, which is a CPLF. The second term is the sum of multiple single-dimensional CPLFs of *different* variables, which is a multi-dimensional CPLF. So, to evaluate this expression, we add the first and second term using the $\oplus$ operator, which, again, results in a CPLF. Second, we use the marginal maximisation operator on this CPLF to obtain the message as required.

Now that we have performed the necessary steps to instantiate the continuous max-sum algorithm, we will compare its performance with that of the discrete max-sum algorithm next.

## 6.3 Empirical Evaluation

To empirically validate the performance of our continuous max-sum algorithm, we focus on its application for coordinating a network of energy constrained wireless sensors deployed for a wide area surveillance task. As noted earlier, we choose this application domain since it represents a challenging real world coordination problem which involves continuously valued control parameters, and also allows us to empirically compare the performance of our novel continuous max-sum algorithm against the conventional discrete

[7] This is equivalent to receiving a message $Q_{x_2 \to f}(x_2) = 0.1x_2$ from variable $x_2$.

one. We note that the discrete max-sum algorithm has previously been benchmarked against approximate (DSA) and complete (DPOP) techniques for DCOPs in a graph k-colouring test-bed (Farinelli et al., 2008), and was shown to obtain better solutions than DSA while scaling very well with the number of sensors, in terms of total message size as opposed to the exponential increment exhibited by DPOP (see Section 2.5.3.1). Therefore, such comparisons are not undertaken here.

In the following, we first describe the problem scenario in detail and then instantiate the sensors' utilities in this setting. We then describe the application of both the conventional discrete max-sum algorithm and our novel continuous max-sum algorithm to this problem, and empirically compare their properties.

### 6.3.1 The Wide Area Surveillance Scenario

We consider a network of wireless sensors that are randomly deployed within some area to detect events (e.g. vehicle and pedestrian activity in an urban setting). We assume that these sensors are able to harvest energy from the environment (e.g. using a photovoltaic cell or vibration-harvesting microgenerators), but at a rate that is insufficient to allow them to be powered continually. Thus at any time a sensor can be in one of two states: either sensing or sleeping. In the former state, the sensor consumes energy at a constant rate, and is able to interact with the surrounding environment (e.g. it can detect events within its sensing field and communicate with other sensors). In the latter state, the sensor cannot interact with the environment but it consumes negligible energy. To maintain energy-neutral operation (Kansal et al., 2007), and thus exhibit an indefinite lifetime, sensors adopt a repeated schedule of length $L$, during which each sensor can be active for only a given time $l_i$. This amount of time depends on specific characteristics of the environment surrounding the sensor, and the means by which energy is harvested. For example, if a sensor is using solar panels to harvest energy, sensors that are in shaded regions will have shorter duty cycles compared to those located in spots with a greater exposure to sunlight.

The sensing ranges of multiple sensors will typically overlap. However, just a single sensor is required to be active in order to detect an event. Thus, there is no gain for the system in having more than one sensor actively sensing the same region (i.e. we have a sub-additive utility function), and hence, to maximise the probability of detecting events while maintaining energy neutral operations, sensors whose sensing fields overlap should coordinate the start times of their duty cycles. Therefore, in this setting, the continuously valued control parameter, $x_i$, represents the time at which sensor $i$ will start sensing, while the domain over which this variable can take values is the interval $[0, L]$. Once the sensors have decided on the value of this parameter, they will repeat this schedule indefinitely.

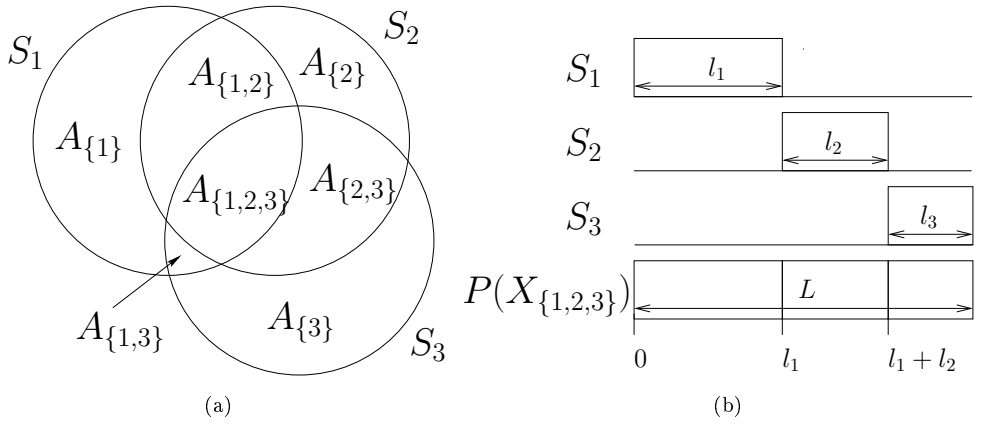



FIGURE 6.5: (a) Example coordination problem in which three sensors, $\{S_1, S_2, S_3\}$, have sensing fields that overlap (b) An optimal solution

### 6.3.2 Applying the Max-Sum Algorithm

In order to apply the max-sum algorithm, in either its continuous or discrete forms, it is first necessary to instantiate the sensors' utility functions for this problem. Thus, we define $\mathbf{N}_i$ to be a set of indexes indicating which other sensors' sensing fields overlap with that of sensor $i$ and $\mathbf{k}$ is any subset of $\mathbf{N}_i$ (including the empty set). $A_{\{i\}\cup\mathbf{k}}$ is the area that is overlapped *only* by sensor $i$ and those sensors in $\mathbf{k}$. For example, with respect to Figure 6.5(a), which shows the three overlapping sensors, the area $A_{\{1,2\}}$ is the area that is sensed only by sensors 1 and 2.[8] With a slight abuse of notation, we represent the entire sensing area of sensor $S_1$ as $S_1$, and thus, note that the area $A_{\{1,2\}}$ is different from $S_1 \cap S_2$ because the area $S_1 \cap S_2$ would include also the sub area $S_1 \cap S_2 \cap S_3$. In general, we have:

$$A_{\{i\}\cup\mathbf{k}} = \bigcap_{j\in(\{i\}\cup\mathbf{k})} S_j \setminus \bigcup_{l\notin(\{i\}\cup\mathbf{k})} S_l \tag{6.6}$$

The utility of sensor $i$ is then simply given by the weighted sum of the probability of detecting an event in any particular sub area:

$$U_i(\mathbf{x}_i) = \sum_{\mathbf{k}\subseteq\mathbf{N}_i} \frac{A_{\{i\}\cup\mathbf{k}}}{|\{i\}\cup\mathbf{k}|} \times P(\mathbf{x}_{\{i\}\cup\mathbf{k}}) \tag{6.7}$$

where $P(\mathbf{x}_{\{i\}\cup\mathbf{k}})$ is the probability of detecting an event per unit area given the combined sensing schedules of sensors $\{i\}\cup\mathbf{k}$. In our experiments, we assume that this is simply given by the fraction of the time during which at least one sensor is actively sensing during

[8] Here, we assume that sensors have knowledge of the overlaps they have with their neighbours. While this may not hold in all applications, we use this simplified model here to focus on the coordination aspects of this application scenario.

the interval of length $L$, and hence, we are assuming that events are instantaneous in time and have no duration. Note, that we divide each sub area by the number of sensors that can sense it to avoid double-counting areas that are represented by multiple sensors. In addition, when the set $\mathbf{k}$ is empty we consider the area covered only by the single sensor. For example, the utility of sensor $S_2$ shown in Figure 6.5(a), is calculated by considering the areas $A_{\{2\}}$, $A_{\{1,2\}}$, $A_{\{2,3\}}$ and $A_{\{1,2,3\}}$.

Given this utility function we can now decompose it into a factor graph on which we run the max-sum algorithm. As discussed in Section 2.5.3.1, there are several ways of doing this. Here, we use a separate function to represent the utility of each sub area and connect this function to all variable nodes that represent sensors whose sensing field overlaps with this sub area. We then remove the functions that model areas where more than two sensors overlap, which is equivalent to only considering pairwise interactions between sensors. This is a very common approach in the DCOP literature, and one that reduces the computational complexity of the coordination, while still providing good solutions in this particular scenario. However, we note that our formalism supports modelling higher order interactions. We can now apply both the discrete and continuous versions of the max-sum algorithm directly to this factor graph.

#### 6.3.2.1 The Discrete Version

To apply the discrete max-sum algorithm, we artificially discretise the continuously valued control parameter and only allow the sensors to select $x_i$ from a set of $d$ discrete values in the range $[0, L]$. The summation and marginal maximisation operators of the max-sum algorithm are performed over this discrete sample space, and the messages exchanged by the sensors are represented by single-dimensional functions evaluated at the $d$ possible sample points.

#### 6.3.2.2 The Continuous Version

To apply the continuous max-sum algorithm, we use the results derived in Section 6.2 to represent the utility functions of the factor graph as continuous piecewise linear functions. For example, the function in Figure 6.1 encodes the utility for two sensors overlapping in a sub area, with $l_1 = 2$, and $l_2 = 5$. The function has a minimum plateau when the active sensing periods of both sensors completely overlap in time (e.g. when $x_1 = x_2 = 0$), and a maximum plateau when sensors do not (e.g. when $x_1 = 0$ and $x_2 \in [2, 5]$).

Notice that if we fix $x_1 = 0$ while increasing $x_2$, the utility increases as well. This is because the length of the interval in which at least one of the two sensor is sensing, increases. When $x_2 = 2$, the function assumes the maximum value and then remains constant until $x_2 = 5$. At this point the function starts decreasing again. This is because

we assume that the sensors compute a schedule for interval $[0, L]$ once, and then use this schedule for intervals $[kL, (k+1)L]$. As a result, when $x_2 > 5$, sensor $S_2$ is sensing during the intervals $[x_2, 10]$ and $[0, 10 - x_2]$, the latter of which overlaps with the sensing interval of $S_1$, thus decreasing the utility.

Consider the following example that illustrates the difference between the discrete and continuous max-sum algorithms. Suppose that sensors are deployed as in Figure 6.5(a), and suppose that to maintain energy neutral operations, the three sensors can be active for $l_1$, $l_2$, and $l_3$ time units out of $L$ (with $l_1 + l_2 + l_3 = L$). In this case, an optimal solution[9] is the one reported in Figure 6.5(b) where $x_1 = 0$, $x_2 = l_1$ and $x_3 = l_1 + l_2$. Notice that, while continuous max-sum is able to assign any value in the interval $[0, L]$ to the sensors' variables, discrete max-sum will be able to find the optimal solution only if the chosen discretisation includes the points $l_1$ and $l_1 + l_2$. However, as previously mentioned, the sensors' duty cycles depend on the sensor deployment and on the environment configuration, and thus they are not known before-hand. Hence, it is not possible to always choose a discretisation that includes the optimal solution and thus discrete max-sum will result in suboptimal solutions.

Given these two algorithm descriptions, we now empirically compare their performance.

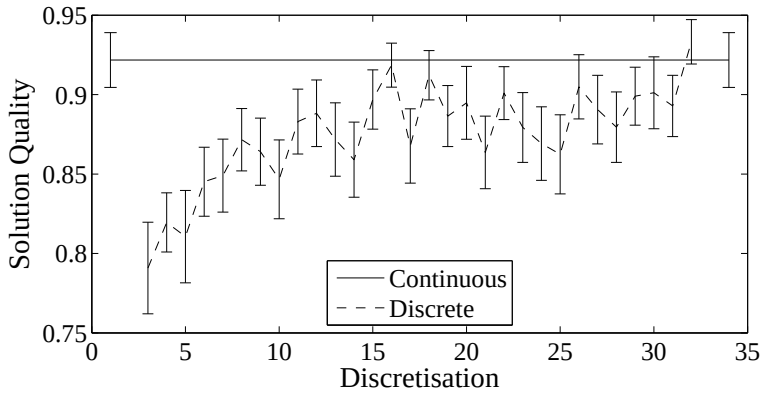
### 6.3.3 Experimental Results

As described above, we performed experiments comparing the performance of discrete and continuous max-sum algorithms to assess the benefits and limitations of our approach. In particular, we compare the two algorithms by considering the quality of the solution and the communication overhead in terms of the size of messages that the sensors must exchange. We performed experiments on deployments of 10 sensors, that are randomly scattered across a unit square. These sensors have a circular shaped sensing area with a radius of 0.2. The sensors' duty cycles $l_i$ are drawn from a uniform distribution over $[0.3, 0.6]$. These values were chosen after initial calibration showed that they produced particularly challenging coordination problems.

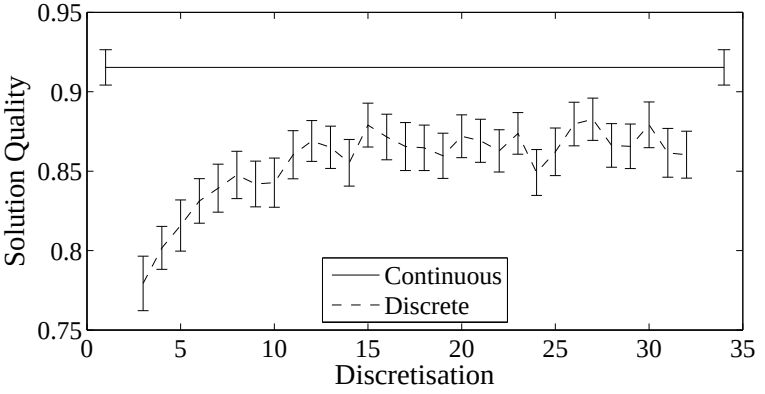
During the experiments we ran both versions of the max-sum algorithm for 20 iterations. Each experiment consisted of a single run of our continuous max-sum algorithm, and multiple runs of the discrete max-sum algorithm with increasing levels of discretisation. Figure 6.6 shows the aggregated results of 100 runs, where the solution quality is expressed in terms of the optimal solution computed with a centralised simulated annealing algorithm.

Specifically, Figure 6.6(a) reports the quality of the final solution (e.g., after the 20 iterations), while Figure 6.6(b) reports the average quality of the solutions obtained after

[9] Note that, in this case, there is an infinite number of optimal solutions, which can be generated by shifting the starting times of all sensors by an equal amount.




(a) Solution quality after 20 iterations




(b) Averaged solution quality over 20 iterations

FIGURE 6.6: Solution quality as a fraction of the optimal solution. Error bars are the standard error in the mean.

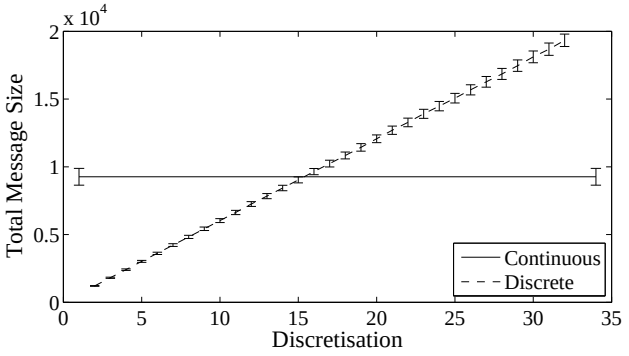



FIGURE 6.7: Total number of values exchanged between the sensors. Error bars are the standard error in the mean.

each iteration. The latter metric incorporates information on how the algorithms behave over time; the quicker the algorithms converge towards better solutions, the higher the average.

Figure 6.6(a) shows that the final solutions produced by continuous max-sum are better than those produced by the discrete version. In particular, continuous max-sum exhibits

up to a 10% increase in the solution quality for low discretisation levels.

Moreover, Figure 6.6(b) shows that, when considering the average quality of solution, the difference is more pronounced also for higher discretisation levels, thus showing that continuous max-sum is able to reach good, stable solutions quicker than the discrete version. This increased performance for the average solution can partially be explained by the faster convergence speed of continuous max-sum: since it takes discrete max-sum longer to converge to a good solution, its solution quality averaged over all 20 iterations is lower than that of continuous max-sum.

In terms of total message size, we can conclude from Figure 6.7 that, as expected, the communication overhead of discrete max-sum increases proportionally to the level of discretisation. Significantly, the continuous max-sum algorithm achieves a better solution quality over the entire range of discretisations, even when the message size of the discrete max-sum algorithm is greater than the continuous version. Thus the continuous version generates better solutions, and also requires less communication overhead.

## 6.4 Conclusions

In this chapter, we presented a novel decentralised algorithm for social welfare maximisation in multi-agent systems, where the agents' control parameters are continuous, and the interactions between the agents is expressible as piecewise linear functions. We extended the existing max-sum algorithm to operate in the domain of continuous variables by using techniques from computational geometry. We empirically evaluated our approach by applying it to a wide area surveillance scenario where energy constrained sensors need to coordinate their sense/sleep schedules to maximise the probability of event detection in a decentralised fashion. We compared the continuous max-sum algorithm with its conventional discrete counterpart, and showed that our continuous algorithm is able to achieve solutions of better quality while exhibiting a lower communication overhead. Clearly, these properties contribute to satisfying the requirements of *accuracy*, and *scalability* that were stated in Section 1.1. Furthermore, since problems that are inherently continuous need no longer be discretised in order to apply a coordination algorithm, this could improve the *modularity* of the system: a larger variety of sensor platforms will be enabled to inter-operate.

Our future work in this area is to extend the proposed approach by considering arbitrary functions that describe the agents' interactions. A promising direction to do this is to investigate the use of techniques such as GPs (Section 2.3.2). Since GPs provide a powerful mathematical tool to approximate arbitrary continuous functions, they could be extremely useful to describe the non-linear interactions between agents. Of course, the key application of such an extension is the mobile sensor problem we have studied throughout this report. We will give more details about future work in Section 7.1.

# Chapter 7

# Conclusions and Future Work

In this report, we have studied the challenge of monitoring environmental phenomena with mobile sensors. In particular, we have looked at previous work on both fixed and mobile wireless sensor networks, both from the perspective of the applications in which they are used, as well as from a more theoretical perspective. Furthermore, we discussed the key concept of adaptive sampling. In particular, adaptive sampling encompasses a set of techniques that aim to maximise the information gain of a sensor network subject to movement constraints, and to the limited resources at the sensors' disposal. We also provided a framework for analysing previous work on adaptive sampling, in which we made a clear distinction between information processing, information valuing, and control algorithms for information maximisation.

In more detail, we showed that the state of the art in adaptive sampling does not meet our requirements of coordinating multiple sensors for environmental monitoring in a robust, scalable, decentralised and modular way. To address these shortcomings, we developed two online decentralised coordination algorithms for monitoring spatial phenomena with mobile sensors. The first operates in un-negotiated mode, in which coordination is achieved exclusively through the exchange of observations. The second algorithm extends the first by adding a principled negotiation stage, based on the max-sum algorithm, in which the sensors negotiate about their moves prior to executing them, which results in negotiated coordination. Both coordination algorithms allow multiple sensors to coordinate their movements and collect highly informative samples from their environment, while at the same time learning the characteristics of that environment. Using different benchmarks, we demonstrated that not only do our algorithms monitor the phenomena better than a network of fixed sensors, but also that they do so with a reduced number of sensors. Also, in contrast to the fixed sensor placement algorithms, our algorithm is capable of learning the characteristics of the environment while monitoring it, without the need for an offline learning phase. Moreover, we showed that the negotiated coordination algorithm outperforms the un-negotiated coordination one, thus showing the effectiveness of explicit coordination between sensors.

Finally, we have demonstrated that our algorithms live up to some, but not all, of the expectations established in the introduction of the report. In particular, we showed that the algorithms are capable of reconstructing the dynamics of the environmental phenomena with reasonable accuracy, that are robust against failures of the sensors, capable of adapting to the environment, and allow the sensors to be autonomous. However, we also concluded that there is still work to be done to put more emphasis on the relevance of observations in terms of the mission goals, tackling more complex environments, and extending the planning horizon of the sensors to consider longer paths. The future work that will aim to address these (and other) issues is discussed in the next section.

## 7.1 Future work

The work described in this report attempts to advance research on coordinating multiple mobile sensors for environmental monitoring, and forms the basis for future work. In particular, we have identified several directions for this future work, ranging from the short-term goals that we will pursue in the upcoming months, to the more general ideas that are considered for the remainder of the time allocated for the PhD. Specifically, the general areas of research that we intend to investigate are:

**Supporting Realistic Environments:** The environments we have considered in the empirical evaluation are relatively easy for the mobile sensors to navigate through. In order to be able to use our work in more realistic scenarios, though, it will be necessary to investigate a way to tackle more complex environmental layouts, in which the sensors' movements are restricted by the type of obstacles that are encountered in realistic scenarios, such as walls, furniture and debris. Moreover, besides physical obstacles, their movement will also be constrained by limited resources, the most important among those being remaining battery life. We believe that individual, greedy control will no longer suffice in these settings, and sensors need to be able to reason about longer (or even infinite) paths (see next item). Furthermore, we want model the sensors in a more realistic way, for instance by defining their action spaces in terms of heading and speed, instead of a discrete movement graph. This will involve extending the continuous max-sum algorithm developed in Chapter 6 for arbitrary continuous utility functions. As mentioned in that chapter, we believe the application of the GP might be a fruitful line of research. Finally, we would like to investigate the applicability of the algorithms developed in this report to the problem of multi-sensor perimeter patrol (e.g. Agmon et al. (2008)), in which strategically acting adversaries try to maximise their intrusion probability.

**Non-Myopic Planning:** In order to tackle these more challenging environments, we need to improve the way in which sensors coordinate. In our current work, the

sensors coordinate by maximising sample value in a myopic fashion (i.e. by considering paths of finite length). We would like to extend our work in this area by applying techniques from sequential decision making (i.e Markov Decision Processes or MDPs), or optimal-control in order to extend the planning horizon of the sensors. In this setting, sensors no longer negotiate over finite plans, but try to maximise the total (discounted) payoff that is obtained over the remaining mission time.

To this end, there are two different solution concepts that we are currently considering. The first involves the approximation of non-myopic planning by combining myopic planning with some (approximate) measure of the advantageousness of the resulting placement. As a result, sensors will not only be incentivised by the immediate reward associated with their finite-length path, but also by the reward obtained by the reduction of uncertainty that is obtained over a longer time. The second solution concept pertains to the notion of stable policies. These are collections of fixed patrolling paths that the sensors follow periodically and whose quality is theoretically bounded. This idea is still in a preliminary stage, but we are interested in the application of the max-sum algorithm on factorised MDPs as put forward by Kok and Vlassis (2005).

**Guaranteeing Network Connectivity** The technique of penalising network disconnection, introduced in Section 5.1.3, is only able to *incentivise* sensors to maintain network connection, not *guarantee* it. The reason for this is that max-sum is not guaranteed to converge on cyclic factor graphs, and consequently, the solution it produces can be of arbitrarily bad quality. For example, such a solution might result in network disconnection, after which the sensors currently have no means of re-establishing communication, since this requires coordination. We wish to investigate techniques that prevent the violation of hard constraints (such as this one), without sacrificing the attractive properties of the max-sum algorithm: scalability, and local computation and communication. As mentioned in Section 5.1.3, we are currently investigating the use of a bounded version of max-sum (Farinelli et al., 2009) that operates on a non-cyclic subgraph of the factor graph. In so doing, it guarantees convergence and gives bounds on the computed solution, at the cost of no longer being able to compute the optimal solution. However, by utilising this extension to the max-sum algorithm, it might be possible to guarantee network connectivity by sacrificing a small amount of information value.

Figure 7.1 summarises our planning of the research activities until the end of the project.

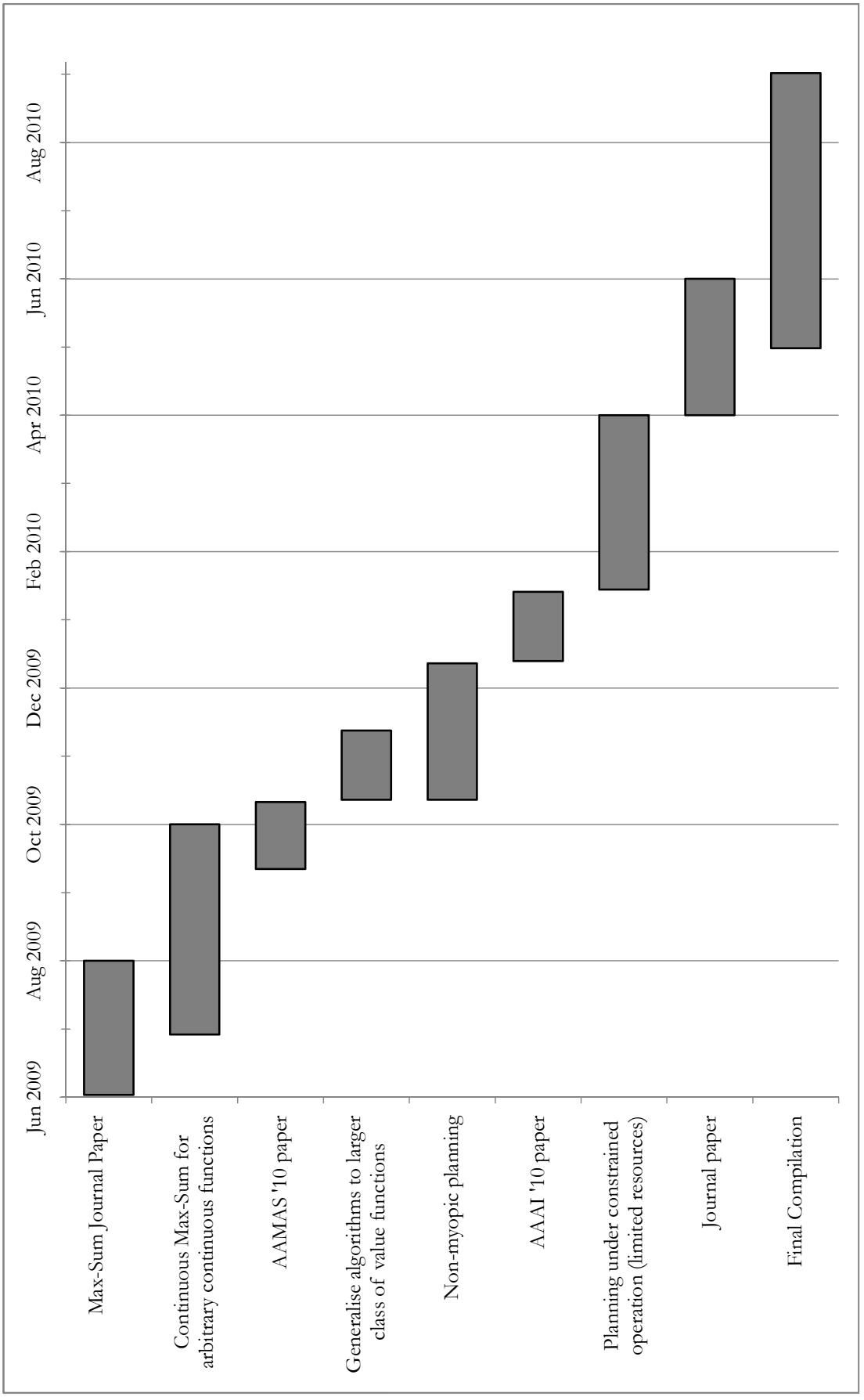



FIGURE 7.1: Gantt chart outlining the planning for the remainder of the project

# Appendix A

# Some Mathematical Background

This appendix briefly covers the mathematical background of the Multivariate Gaussian distribution on which the theory of the GP is based and some of the information metrics that are commonly used in adaptive sampling.

## A.1 Multivariate Gaussian distribution

This appendix summarises the main properties of the multivariate Gaussian distribution, which are used in Section 2.3.2.

**Probability distribution** If elements $y_1, \ldots, y_D$ of a $D$ dimensional column vector $\mathbf{y}$ are jointly Gaussian with mean $\boldsymbol{\mu}$, and covariance $\boldsymbol{\Sigma}$ their pdf is given by:

$$p(\mathbf{y}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{D/2}\,|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{y}-\boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\mathbf{y}-\boldsymbol{\mu})\right) \tag{A.1}$$

This notation is commonly abbreviated by $p(\mathbf{y}|\mu, \boldsymbol{\Sigma}) = \mathcal{N}(\mu, \Sigma)$, or $\mathbf{y} \sim \mathcal{N}(\mu, \Sigma)$.

**Marginal distribution** If the joint probability distribution of the vector $\begin{bmatrix}\mathbf{y_1} & \mathbf{y_2}\end{bmatrix}^\top$ is given by:

$$\begin{bmatrix}\mathbf{y_1}\\ \mathbf{y_2}\end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix}\boldsymbol{\mu_1}\\ \boldsymbol{\mu_2}\end{bmatrix}, \begin{bmatrix}A & C\\ C^\top & B\end{bmatrix}\right), \tag{A.2}$$

$\mathbf{y_1}$ has a Gaussian distribution with parameters $\boldsymbol{\mu_1}$ and $A$: $p(\mathbf{y_1}) = \mathcal{N}(\boldsymbol{\mu_1}, A)$. In other words, both the marginal probability distributions of $\mathbf{y_1}$ and $\mathbf{y_2}$ are Gaussian.

**Conditional distribution** In general the conditional distribution of $\mathbf{y_1}$ given $\mathbf{y_2}$ is:

$$P(\mathbf{y_1}|\mathbf{y_2}) = \frac{P(\mathbf{y_1}, \mathbf{y_2})}{P(\mathbf{y_2})} \tag{A.3}$$

When $\mathbf{y_1}$ given $\mathbf{y_2}$ are jointly Gaussian as in Equation their joint probability distribution is given by Equation A.2, the distribution of $\mathbf{y_1}$ conditioned on $\mathbf{y_2}$ is:

$$\mathbf{y_1}|\mathbf{y_2} \sim \mathcal{N}\left(\mu_\mathbf{1} + CB^{-1}(\mathbf{y_2} - \boldsymbol{\mu_2}), A - CB^{-1}C^\top\right) \tag{A.4}$$

## A.2 Information metrics

This appendix gives a brief overview of the mathematical details of the information metrics described in Section 2.4.

The surprise or self-information $I(\omega_n)$ of an outcome $\omega_n$ of a discrete random variable $X$ is defined as:

$$I(\omega_n) = -\log p(X = \omega_n) \tag{A.5}$$

The Shannon information content $h(\omega_n)$ of an outcome $\omega_n$ is similar to self-information, but is measured in base-2, and the units are referred to as *bits*:

$$h(\omega_n) = -\log_2 p(X = \omega_n) \tag{A.6}$$

Now, differential entropy $H(X)$ is the expected value of the self-information of $X$:

$$H(X) = E[I(\omega_n)] = -\int_{-\infty}^{\infty} \log(p(x))p(x)dx \tag{A.7}$$

Mutual information $MI(X;Y)$ is the decrease in entropy in random variable $X$, given $Y$, and *vice versa*:

$$MI(X;Y) = H(X) - H(X|Y) = H(Y) - H(Y|X) \tag{A.8}$$

Informally, this can be thought of as the information $X$ and $Y$ have in common.

Finally, Kullback-Leibler divergence (or relative entropy) measures the difference between two probability distributions $P$ and $Q$:

$$D_{\mathsf{KL}}(P\|Q) = \int_{-\infty}^{\infty} p(x) \log \frac{p(x)}{q(x)} dx \tag{A.9}$$

# Bibliography

N. Agmon, V. Sadov, G. A. Kaminka, and S. Kraus. The impact of adversarial knowledge on adversarial planning in perimeter patrol. In *Proceedings of the 7th International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS 2008)*, pages 55–62, Estoril, Portugal, 2008.

M. Ahmadi and P. Stone. A multi-robot system for continuous area sweeping tasks. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1724–1729, 2006.

S. M. Aji and R. J. McEliece. Generalized distributive law. *IEEE Transactions on Information Theory*, 46(2):325–343, 2000.

G. Arslan, J. R. Marden, and J. S. Shamma. Autonomous vehicle-target assignment: A game theoretical formulation. *ASME Journal of Dynamic Systems, Measurement and Control*, 129(5):584–596, 2007.

D. Calisi, A. Farinelli, L. Iocchi, and D. Nardi. Autonomous exploration for search and rescue robots. In *WIT Transactions on the Built Environment*, volume 94, pages 305–314, Malta, 2007.

R. K. Dash, A. Rogers, N. R. Jennings, S. Reece, and S. Roberts. Constrained bandwidth allocation in multi-sensor information fusion: a mechanism design approach. In *Proceedings of the 7th International Conference on Information Fusion (FUSION)*, Philadelphia, Pennsylvania, USA, 2005.

A. Deshpande, C. Guestrin, and S. Madden. Resource-aware wireless sensor-actuator networks. *IEEE Data Engineering Bulletin*, 28(1):40–47, 2005a.

A. Deshpande, C. Guestrin, S. Madden, J. M. Hellerstein, and W. Hong. Model-based approximate querying in sensor networks. *International Journal on Very Large Data Bases*, 14(4):417–443, 2005b.

A. Dhariwal, B. Zhang, C. Oberg, B. Stauffer, A. Requicha, D. Caron, and G.S. Sukhatme. Networked aquatic microbial observing system. In *Proceedings of the 2006 Conference on International Robotics and Automation*, pages 4285–4287, Orlando, Florida, USA, 2006.

P. Dunias. Robot motion planning using potential fields. In *Symposium on Robotics and Cybernetics. CESA '96 IMACS Multiconference. Computational Engineering in Systems Applications*, pages 611–616, Lille, France, 1996.

H. F. Durrant-Whyte, B. Y. S. Rao, and H. Hu. Toward a fully decentralized architecture for multi-sensor data fusion. In *Proceedings of the IEEE International Conference on Robotics and Automation*, volume 2, pages 1331–1336, Los Alamitos, California, USA, 1990.

C. Edordu and L. Sacks. Self organising wireless sensor networks as a land management tool in developing countries: A preliminary survey. Technical report, University College London, 2006.

J. Elson and D. Estrin. Sensor networks: A bridge to the physical world. In C.S. Raghavendra, Krishna M. Sivalingam, and Taieb Znati, editors, *Wireless Sensor Networks*, chapter 1. Springer, 2004.

M. R. Endsley. Toward a theory of situation awareness in dynamic systems. *Human Factors*, 37(1):32–64, 1995.

A. Farinelli, A. Rogers, and N. R. Jennings. Bounded approximate decentralised coordination using the max-sum algorithm. In *Proceedings of the IJCAI 2009 Workshop on Distributed Constraint Reasoning (DCR)*, Pasadena, California, USA, 2009.

A. Farinelli, A. Rogers, A. Petcu, and N. R. Jennings. Decentralised coordination of low-power embedded devices using the max-sum algorithm. In *Proceedings of the 7th International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS 2008)*, pages 639–646, Estoril, Portugal, 2008.

S. Fitzpatrick and L. Meertens. Distributed coordination through anarchic optimization. In Victor Lesser, Charles L. Ortiz, Jr., and Milind Tambe, editors, *Distributed Sensor Networks*, chapter 11, pages 257–295. Kluwer Academic Publishers, 2003.

B. J. Frey and D. Dueck. Clustering by passing messages between data points. *Science*, 315(5814):972–976, February 2007.

T. Fujiwara, H. Makie, and T. Watanabe. A framework for data collection system with sensor networks in disaster circumstances. In *Proceedings of the International Workshop on Wireless Ad-Hoc Networks*, pages 94–98, Oulu, Finland, 2004.

S. Galmes. Lifetime issues in wireless sensor networks for vineyard monitoring. In *Proceedings of the IEEE International Conference on Mobile Adhoc and Sensor Systems (MASS)*, pages 542–545, Vancouver, Canada, 2006.

J. Grace and J. Baillieul. Stochastic strategies for autonomous robotic surveillance. In *Proceedings of the 44th IEEE Conference on Decision and Control, and the European*

*Control Conference (CDC-ECC 2005)*, volume 2005, pages 2200–2205, Seville, Spain, 2005.

B. Grocholsky. *Information-Theoretic Control of Multiple Sensor Platforms*. PhD thesis, University of Sydney, 2002.

B. Grocholsky, J. Keller, V. Kumar, and G. Pappas. Cooperative air and ground surveillance. *IEEE Robotics & Automation Magazine*, 13(3):16–25, 2006.

B. Grocholsky, A. Makarenko, T. Kaupp, and H. F. Durrant-Whyte. Scalable control of decentralised sensor platforms. In *Proceedings of the second International Workshop on Information Processing in Sensor Network (IPSN 2003)*, pages 96–112. Springer Berlin / Heidelberg, 2003.

B. Grocholsky, R. Swaminathan, J. Keller, V. Kumar, and G. Pappas. Information driven coordinated air-ground proactive sensing. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 2211–2216, Barcelona, Spain, 2005.

J. Gross and J. Yellen. *Graph theory and its applications*. CRC Press, Inc., Boca Raton, Florida, USA, 1999.

C. Guestrin, A. Krause, and A. P. Singh. Near-optimal sensor placements in gaussian processes. In *Proceedings of the 22nd International Conference on Machine Learning (ICML)*, pages 265–272, New York, New York, USA, 2005. ACM Press.

P. Guttorp and P. D. Sampson. Methods for estimating heterogeneous spatial covariance functions with environmental applications. *Handbook of Statistics*, 12:661–689, 1994.

J. K. Hart and K. Martinez. Environmental sensor networks: A revolution in the earth system science? *Earth-Science Reviews*, 78(3-4):177–191, 2006.

J. K. Hart, K. Martinez, R. Ong, A. Riddoch, K. C. Rose, and P. Padhy. A wireless multi-sensor subglacial probe: Design and preliminary results. *Journal of Glaciology*, 51(178):389–397, 2006.

J. Hershberger. Finding the upper envelope of $n$ line segments in $O(n \log n)$ time. *Information Processing Letters*, 33(4):169–174, 1989.

S.J. Julier and J.K. Uhlmann. A non-divergent estimation algorithm in the presence of unknown correlations. In *Proceedings of the American Control Conference*, volume 4, pages 2369–2373, 1997.

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

Y. Kato and T. Mukai. A new method of localizing gas source positions for a mobile robot with gas sensors. In *The 13th International Conference on Solid-State Sensors, Actuators and Microsystems. Digest of Technical Papers*, volume 2, pages 2155–8, Seoul, South Korea, 2005.

W. Kerr and D. Spears. Robotic simulation of gases for a surveillance task. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2905–2910, Edmonton, Alberta, Canada, 2005.

J. Kho, R. Rogers, and N. R. Jennings. Decentralised control of adaptive sampling in wireless sensor networks. *ACM Transactions on Sensor Networks*, 5(3), March 2009. In Press.

P. Klapwijk. TICS:a topology based infrastructure for crisis situations. Master's thesis, Delft University of Technology, 2005.

C. W. Ko, J. Lee, and M. Queyranne. An exact algorithm for maximum entropy sampling. *Operations Research*, 43(4):684–691, 1995.

J. R. Kok and N. Vlassis. Using the max-plus algorithm for multiagent decision making in coordination graphs: extended abstract. In *Proceedings of the 17th Belgian-Dutch Conference on Artifical Intelligence (BNAIC)*, pages 359–360, Brussels, Belgium, 2005.

A. Krause and C. Guestrin. Near-optimal nonmyopic value of information in graphical models. In *Proceedings of the 21th Annual Conference on Uncertainty in Artificial Intelligence (UAI-05)*, pages 324–33, Arlington, Virginia, USA, 2005.

A. Krause and C. Guestrin. Nonmyopic active learning of gaussian processes: an exploration-exploitation approach. In *Proceedings of the 24th International Conference on Machine Learning*, pages 449–456. ACM Press, New York, New York, USA, 2007. ISBN 978-1-59593-793-3.

A. Krause, C. Guestrin, A. Gupta, and J. Kleinberg. Near-optimal sensor placements: Maximizing information while minimizing communication cost. In *Proceedings of the 5th international conference on Information Processing in Sensor Networks*, pages 2–10, New York, NY, USA, 2006. ACM Press.

F. R. Kschischang, B. J. Frey, and H. A. Loeliger. Factor graphs and the sum-product algorithm. *IEEE Transactions on Information Theory*, 47(2):498–519, 2001.

N. Kurata, S. Saruwatari, and H. Morikawa. Ubiquitous structural monitoring using wireless sensor networks. In *Proceedings of the International Symposium on Intelligent Signal Processing and Communications*, pages 99–102, Yonago, Japan, 2006.

A. LaMarca, W. Brunette, D. Koizumi, M. Lease, S.B. Sigurdsson, K. Sikorski, D. Fox, and G. Borriello. Making sensor networks practical with robots. In *Proceedings of*

*First International Conference on Pervasive Computing*, volume 2414 of *Lecture Notes in Computer Science*, pages 152–66, Zurich, Switzerland, 2002.

K. Langendoen, A. Baggio, and O. Visser. Murphy loves potatoes: experiences from a pilot sensor network deployment in precision agriculture. In *Proceedings of the 14th International Workshop on Parallel and Distributed Real-Time Systems (WPDRTS)*, Rhodes, Greece, 2006.

A. Lilienthal, D. Reiman, and A. Zell. Gas source tracing with a mobile robot using an adapted moth strategy. In Informatik aktuell, editor, *Autonome Mobile Systeme (AMS)*, pages 150–160, Stuttgart, Germany, 2003.

K. H. Low, J. M. Dolan, and P. Khosla. Adaptive multi-robot wide-area exploration and mapping. In *Proceedings of the 7th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS 2008)*, pages 23–30, Estoril, Portugal, 2008.

D. J. C. MacKay. *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press, 2003.

R. J. Maheswaran, J. Pearce, and M. Tambe. A family of graphical-game-based algorithms for distributed constraint optimization problems. In *Coordination of Large-Scale Multiagent Systems*, pages 127–146. Springer-Verlag, Heidelberg Germany, 2005.

R. Mailler and V. Lesser. Solving distributed constraint optimization problems using cooperative mediation. In *Proceedings of Third International Joint Conference on Autonomous Agents and MultiAgent Systems (AAMAS 2004)*, pages 438–445, New York, New York, USA, 2004.

A. Makarenko and H. Durrant-Whyte. Decentralized bayesian algorithms for active sensor networks. *Information Fusion*, 7(4):418–33, 2006.

S. Mandal, D. Saha, and T. Banerjee. A neural network based prediction model for flood in a disaster management system with sensor networks. In *Proceedings of the International Conference on Intelligent Sensing and Information Processing*, pages 78–82, Chennai, India, 2005.

A. Meliou, A. Krause, C. Guestrin, and J. M. Hellerstein. Nonmyopic informative path planning in spatio-temporal models. In *Proceedings of the 22nd Conference on Artificial Intelligence (AAAI)*, pages 602–607, 2007.

J. P. Modi, P. Scerri, W. Shen, and M. Tambe. Distributed resource allocation. In Victor Lesser, Charles L. Ortiz, Jr., and Milind Tambe, editors, *Distributed Sensor Networks*, chapter 10, pages 219–256. Kluwer Academic Publishers, 2003.

J. P. Modi, M. Tambe, and M. Yokoo. ADOPT: Asynchronous distributed constraint optimization with quality guarantees. *Artificial Intelligence*, 161:149–180, 2005.

R. Murphy, J. Casper, J. Hyams, M. Micire, and B. Minten. Mobility and sensing demands in usar. In *Proceedings of the 26th Annual Conference of the IEEE Industrial Electronics Society. (IECON 2000)*, volume 1, pages 138–42, Nagoya, Japan, 2000.

G. L. Nemhauser and L. A. Wolsey. An analysis of approximations for maximising submodular set functions—I. *Mathematical Programming*, 14(1):265—294, 1978.

H. G. Nguyen, N. Farrington, and N. Pezeshkian. Maintaining communication link for tactical ground robots. In *AUVSI's Unmanned Systems North America 2004 - Proceedings*, pages 311–323, Anaheim, California, United States, 2004.

D. J. Nott and W. T. M. Dunsmuir. Estimation of nonstationary spatial covariance structure. *Biometrika*, 89:819–829, 2002.

M. A. Osborne, A. Rogers, S. D. Ramchurn, S. J. Roberts, and N. R. Jennings. Towards real-time information processing of sensor network data using computationally efficient multi-output gaussian processes. In *Proceedings of the 7th International Conference on Information Processing in Sensor Networks (IPSN 2008)*, pages 109–120, St. Louis, Illinois, USA, 2008.

P. Padhy, R. K. Dash, K. Martinez, and N. R. Jennings. Utility-based adaptive sensing and multi-hop communication protocol for wireless sensor networks. Technical report, University of Southampton, 2007.

M. Paskin, C. Guestrin, and J. McFadden. A robust architecture for distributed inference in sensor networks. In *Proceedings of the 4th International Symposium on Information Processing in Sensor Networks (IPSN 2005)*, page 8, Piscataway, NJ, USA, 2005. ISBN 0-7803-9202-7.

G. A. S. Pereira, M. B. Soares, and M. F. M. Campos. A potential field approach for collecting data from sensor networks using mobile robots. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, volume 4, pages 3469–3474, Sendai, Japan, 2004.

A. Petcu and B. Faltings. A scalable method for multiagent constraint optimization. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI 2005)*, pages 266–271, Edinburgh, Scotland, 2005.

R. Pon, M.A. Batalin, V. Chen, A. Kansal, Duo Liu, M. Rahimi, L. Shirachi, A. Somasundra, Yan Yu, M. Hansen, W.J. Kaiser, M. Srivastava, G. Sukhatme, and D. Estrin. Coordinated static and mobile sensing for environmental monitoring. In *Distributed Computing in Sensor Systems. First IEEE International Conference, DCOSS 2005. Proceedings (Lecture Notes in Computer Science Vol. 3560)*, pages 403–405, Marina del Rey, CA, USA, 2005a.

R. Pon, A. Kansal, D. Liu, M. Rahimi, L. Shirachi, W. J. Kaiser, G. J. Pottie, M. Srivastava, G. Sukhatme, and D. Estrin. Networked Infomechanical Systems (NIMS): Next generation sensor networks for environmental monitoring. In *IEEE MTT-S International Microwave Symposium Digest*, volume 2005, pages 373–376, Long Beach, California, United States, 2005b.

M. Rahimi, R. Pon, W. J. Kaiser, G. S. Sukhatme, D. Estrin, and M. Srivastava. Adaptive sampling for environmental robotics. In *Proceedings of the IEEE International Conference on Robotics and Automation*, volume 4, pages 3537–3544, New Orleans, Louisiana, USA, 2004.

C. E. Rasmussen and Z. Ghahramani. Bayesian Monte Carlo. In Sebastian Thrun Suzanna Becker and Klaus Obermayer, editors, *Advances in Neural Information Processing Systems*, volume 15, pages 489–496. MIT Press, 2003.

C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, 2006.

S. Reece and S. Roberts. Robust, low-bandwidth, multi-vehicle mapping. In *Proceedings of the 8th International Conference on Information Fusion (Fusion 2005)*, volume 2, 2005.

S. Reece and S. Roberts. Information fusion and inference in dynamic inference environments. Technical Report D1.1-11, Oxford University, Department of Engineering Science, October 2008.

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

N. Roy, W. Burgard, D. Fox, and S. Thrun. Coastal navigation - mobile robot navigation with uncertainty in dynamic environments. In *Proceedings of the IEEE International Conference on Robotics and Automation*, volume 1, pages 35–40, Detroit, Michigan, USA, 1999.

K. Sha, W. Shi, and O. Watkins. Using wireless sensor networks for fire rescue applications: requirements and challenges. In *Proceedings of the IEEE International Conference on Electro/Information Technology*, East Lansing, Michigan, USA, 2006.

A. Singh, A. Krause, C. Guestrin, W. J. Kaiser, and M. A. Batalin. Efficient planning of informative paths for multiple robots. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2204–2211, Hyderabad, India, 2007.

C. Stachniss, G. Grisetti, and W. Burgard. Information gain-based exploration using rao-blackwellized particle filters. In *Proceedings of Robotics: Science and Systems (RSS)*, pages 65–72, 2005.

M. L. Stein. *Interpolation of Spatial Data*. Springer, 1999.

R. Stranders, A. Farinelli, A. Rogers, and N. R. Jennings. A decentralized, on-line coordination mechanism for monitoring spatial phenomena with mobile sensors. In *Proceedings of the Second International Workshop on Agent Technology for Sensor Networks (ATSN)*, pages 9–15, Estoril, Portugal, 2008.

R. Stranders, A. Farinelli, A. Rogers, and N. R. Jennings. Decentralised Coordination of Continuously Valued Control Parameters using the Max-Sum Algorithm. In *Proceedings of the 8th International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS 2009)*, pages 601–608, Budapest, Hungary, 2009a.

R. Stranders, A. Farinelli, A. Rogers, and N. R. Jennings. Decentralised coordination of mobile sensors using the max-sum algorithm. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI 2009)*, Pasadena, California, USA, 2009b.

R. Szewczyk, E. Osterweil, J. Polastre, M. Hamilton, A. Mainwaring, and D. Estrin. Habitat monitoring with sensor networks. *Communications of the ACM*, 47(6):34–40, 2004.

J. E. Vargas, K. Tvalarparti, and Z. Wu. Target tracking with bayesian estimation. In V. Lesser, C. L. Ortiz, Jr., and M. Tambe, editors, *Distributed Sensor Networks*, chapter 5, pages 219–256. Kluwer Academic Publishers, 2003.

T. Wark, P. Corke, P. Sikka, L. Klingbeil, Ying Guo, C. Crossman, P. Valencia, D. Swain, and G. Bishop-Hurley. Transforming agriculture through pervasive wireless sensor networks. *IEEE Pervasive Computing*, 6(2):50–57, 2007.

Y. Weiss and W. T. Freeman. On the optimality of solutions of the max-product belief propagation algorithm in arbitrary graphs. *IEEE Transactions on Information Theory*, 47(2):723–735, 2001.

B. Zhang and G. S. Sukhatme. Adaptive sampling for estimating a scalar field using a robotic boat and a sensor network. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 3673–3680, Roma, Italy, 2007.

T. Zhang, S. Madhani, and E. van den Berg. Sensors on patrol (SOP): using mobile sensors to detect potential airborne nuclear, biological, and chemical attacks. In *Proceedings of the IEEE Military Communications Conference (MILCOM)*, volume 5, pages 2924–2929, Atlantic City, New Jersey, USA, 2005.

Z. Zhang. Investigation of wireless sensor networks for precision agriculture. In *Proceedings of the ASAE Annual International Meeting 2004*, pages 1157–1164, Ottawa, Ontario, Canada, 2004.

J. Zhou, D. De Roure, and S. Vivekanandan. Adaptive sampling and routing in a floodplain monitoring sensor network. In *Proceedings of the 2nd IEEE International Conference on Wireless and Mobile Computing Networking and Communications*, Montreal, Quebec, Canada, 2006.