UNIVERSITY OF SOUTHAMPTON
Faculty of Physical and Applied Sciences
Electronics and Computer Science

Theory and Practice of Coordination Algorithms exploiting the Generalised Distributive Law

by Francesco M. Delle Fave

Supervisors: Professor Nicholas R. Jennings and Doctor Alex Rogers
Examiners: Doctor Niki Trigoni and Professor Sandor M. Veres

A thesis submitted in partial fulfilment for the
degree of Doctor of Philosophy

September 2012

UNIVERSITY OF SOUTHAMPTON

ABSTRACT

Faculty of Physical and Applied Sciences
ELECTRONICS AND COMPUTER SCIENCE

Doctor of Philosophy

by Francesco M. Delle Fave

A key challenge for modern computer science is the development of technologies that allow interacting computer systems, typically referred as agents, to coordinate their decisions whilst operating in an environment with minimal human intervention. By so doing, the decision making capabilities of each of these agents should be improved by making decisions that take into account what the remaining agents intend to do. Against this background, the focus of this thesis is to study and design new coordination algorithms capable of achieving this improved performance.

In this line of work, there are two key research challenges that need to be addressed. First, the current state-of-the-art coordination algorithms have only been tested in simulation. This means that their practical performance still needs to be demonstrated in the real world. Second, none of the existing algorithms are capable of solving problems where the agents need to coordinate over complex decisions which typically require to trade off several parameters such as multiple objectives, the parameters of a sufficient statistic and the sample value and the bounds of an estimator. However, such parameters typically characterise the agents' interactions within many real world domains. For this reason, deriving algorithms capable of addressing such complex interactions is a key challenge to bring research in coordination algorithms one step closer to successful deployment.

The aim of this thesis is to address these two challenges. To achieve this, we make two types of contribution. First, we develop a set practical contributions to address the challenge of testing the performance of state-of-the-art coordination algorithms in the real world. More specifically, we perform a case study on the deployment of the max-sum algorithm, a well known coordination algorithm, on a system that is couched in terms of allowing the first responders at the scene of a disaster to request imagery collection tasks of some of the most relevant areas to a team of unmanned aerial vehicles (UAVs). These agents then coordinate to complete the largest number of tasks. In more detail, max-sum is based on the generalised distributive law (GDL), a well known algebraic framework that has been used in disciplines such as artificial intelligence, machine learning and statistical physics, to derive effective algorithms to solve optimisation problems. Our

contribution is the deployment of max-sum on real hardware and the evaluation of its performance in a real world setting. More specifically, we deploy max-sum on two UAVs (hexacopters) and test it a number of different settings. These tests show that max-sum does indeed perform well when confronted with the complexity and the unpredictability of the real world.

The second category of contributions are theoretical in nature. More specifically, we propose a new framework and a set of solution techniques to address the complex interactions requirement. To achieve this, we move back to theory and tackle a new class of problem involving agents engaged in complex interactions defined by multiple parameters. We name this class partially ordered distributed constraint optimisation problems (PO-DCOPs). Essentially, this generalises the well known distributed constraint optimisation problem (DCOP) framework to settings in which agents make decisions over multiple parameters such as multiple objectives, the parameters of a sufficient statistic and the sample value and the bounds of an estimator. To measure the quality of these decisions, it becomes necessary to strike a balance between these parameters and to achieve this, the outcome of these decisions is represented using partially ordered constraint functions.

Given this framework, we present three sub-classes of PO-DCOPs, each focusing on a different type of complex interaction. More specifically, we study (i) multi-objective DCOPs (MO-DCOPs) in which the agents' decisions are defined over multiple objectives, (ii) risk-aware DCOPs (RA-DCOPs) in which the outcome of the agents' decisions is not known with certainty and thus, where the agents need to carefully weigh the risk of making decisions that might lead to poor and unexpected outcomes and, (iii) multi-arm bandit DCOPs (MAB-DCOPs) where the agents need to learn the outcome of their decisions online. To solve these problems, we again exploit the GDL framework. In particular, we employ the flexibility of the GDL to obtain either optimal or bounded approximate algorithms to solve PO-DCOPs. The key insight is to use the algebraic properties of the GDL to instantiate well known DCOP algorithms such as DPOP, Action GDL or bounded max-sum to solve PO-DCOPs. Given the properties of these algorithms, we derive a new set of solution techniques. To demonstrate their effectiveness, we study the properties of these algorithms empirically on various instances of MO-DCOPs, RA-DCOPs and MAB-DCOPs. Our experiments emphasize two key traits of the algorithms. First, bounded approximate algorithms perform well in terms of our requirements. Second, optimal algorithms incur an increase in both the computation and communication load necessary to solve PO-DCOPs because they are trying to optimally solve a problem which is potentially more complex than canonical DCOPs.

Contents

List of Figures

List of Tables

List of Algorithms

Declaration of Authorship

I, Francesco Maria Delle Fave, declare that the thesis entitled *Theory and Practice of Coordination Algorithms exploiting the Generalised Distributive Law* and the work presented in the thesis are both my own, and have been generated by me as the result of my own original research. I confirm that:

- this work was done wholly or mainly while in candidature for a research degree at this University;
- where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated;
- where I have consulted the published work of others, this is always clearly attributed;
- where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work;
- I have acknowledged all main sources of help;
- where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself;
- parts of this work have been published in a number of conference and journal papers (see Section 1.2 for a detailed list).

Signed:.....

Date:.....

Acknowledgements

There are numerous people which contributed to the development of this thesis and which I would like to thank.

First and foremost, I need to express my deepest gratitude to my supervisors Nick Jennings and Alex Rogers. In particular, I would like to thank them for their insights that guided me towards understanding how to transform intuitions into concrete ideas and ideas into actual research. By so doing, I acquired an invaluable experience which will be very useful in the future. Moreover, I would like to thank them for giving me the opportunity to visit the Australian Centre for Field Robotics at the University of Sydney. It was a tough but great and successful experience, which taught me how to push forward my skills and limits.

Second, I would like to thank all the co-authors of the papers I have written. In particular, I would like to thank doctor Ruben Stranders for his help on building the work on partially ordered distributed constraint optimisation problems. Next, I would like to thank Zhe Xu for teaching me about unmanned aerial vehicles, doctor Long Tran-Thanh for helping me to understand the theory behind multi-armed bandits and doctor Alessandro Farinelli for his insights on the max-sum algorithm and on the generalised distributive law in general.

Next, I would like to thank all the colleagues that I met at the Agents, Interactions and Complexity Group during these three years. In particular, I would like to thank Ruben Stranders a colleague and a friend, who helped me a lot by giving me good advices and constructive criticism. I would then like to thank Simon Williamson, Archie Chapman, Sebastian Stein, Long Tran-Thanh and Ramachandra Kota for their advices and their help during these three years.

This research was funded by the ALADDIN project. Hence, I would like to thank the different sponsors of the project for giving me the opportunity to pursue a PhD in computer science at the University of Southampton.

Finally, I would like to thank my family and friends for their help and support. They all came to visit me multiple times in these three years and it was a great help and distraction in the days where I was working hard. To them is dedicated this thesis.

Acronyms

DCOP	Distributed Constraint Optimisation Problem
COP	Constraint Optimisation Problem
PO-DCOP	Partially-Ordered Distributed Constraint Optimisation Problem
MO-DCOP	Multi-Objective Distributed Constraint Optimisation Problem
RA-DCOP	Risk-Aware Distributed Constraint Optimisation Problem
MAB-DCOP	Multi-Armed Bandit Distributed Constraint Optimisation Problem
DPOP	Dynamic Programming Optimisation Protocol
MAB	Multi-Armed Bandit
GDL	Generalised Distributive Law
PO-GDL	Partially-Ordered Generalised Distributive Law
PDS	Pareto-Dominated / Sum
NDC	Non-Dominated / Convolution
UCB	Upper Confidence Bound
DSA	Distributed Stochastic Algorithm
DNE	Distributed Neighbour Exchange Algorithm
MGM	Maximum Gain Messaging
UAV	Unmanned Aerial Vehicle
UGV	Unmanned Ground Vehicle
AUV	Autonomous Underwater Vehicle
pdf	Probability Density Function
PDA	Personal Digital Assistant
SLAM	Simultaneous Localisation and Mapping
GPS	Global Positioning System
EUS	Expectation of the Utility of the Sum of Local Constraints
SEU	Sum of the Expected Utilities of the Local Constraints

Nomenclature

A	The set of agents of a DCOP
X	The set of variables of a DCOP
x_i	One variable in X
D	The set of the domains of the variables X of a DCOP
D_i	The domain of variable x_i
C	The set of constraint functions of a DCOP
C_j	One constraint function in C
F	The function of a DCOP assigning variables in X to agents in A
U	The utility function representing the risk profile of a RA-DCOP
m	The number of a DCOP's constraint functions
n	The number of a DCOP's variables
k	The number of objectives of a MO-DCOP
$Q_{i\to j}(x_i)$	A variable to function message in the GDL framework
$R_{j\to i}(x_i)$	A function to variable message in the GDL framework
$\otimes$	The aggregation operator in the GDL framework
$\oplus$	The counting operator in the GDL framework
ND_X	The set of non-dominated values of a PO-DCOP
O_X	The set of solutions corresponding to the set of non-dominated values ND of a PO-DCOP
$\succ$	the binary algebraic relation defining a partial order in a PO-DCOP
$\max_{\succ}$	the operator used in all PO-DCOP to calculate the non-dominated values in ND
V	The utopia point of a multi-objective optimisation problem
$\mathcal{T}$	The set of tasks in the dynamic task allocation problem presented in Chapter 2

T_j	An imagery collection task in $\mathcal{T}$
X^*	The optimal assignment to the variables.
w_{ij}	A weight of the edge of a factor graph between variable x_i and function C_j (see Chapter 2)
w_{ij}^k	A weight of an edge of the factor graph related to objective k in a MO-DCOP
$\mathbf{w}_{ij}$	A vector of weights w_{ij}^k (Chapter 2)
$\mathbf{W}$	The sum of all the weights $\mathbf{w}_{ij}$ pruned from the factor graph

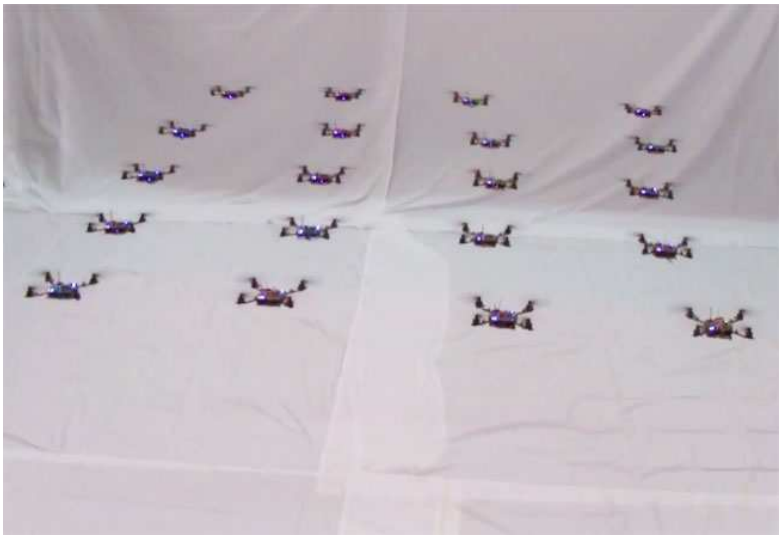
Chapter 1

Introduction

The development of technologies that allow interacting computer systems to operate collectively, as a team, in an environment with minimal human intervention, has become a key challenge for current computer science research that sits at the intersection of artificial intelligence, robotics and machine learning. Each of the individual components that make up these systems typically has to perceive, reason and act within a dynamic environment and, for this reason, they are often referred to as autonomous agents (Jennings, 2001).

Among these technologies, particular attention has been given to coordination techniques in which the decision making process of each autonomous agent is improved by taking into account the decisions of those other agents with whom it interacts (i.e. its neighbours). In so doing, the team agrees on a *collective* decision that represents the joint set of their individual decisions. For instance, by adopting a coordinated behaviour, each agent improves its performance since it is aware of what each other member of the team is going to do. Hence, it can discard redundant or sub-optimal decisions (e.g. another agent is doing the same or it is doing something irrelevant to the operation) that would decrease the performance of the team if selected. As a consequence, the performance of the entire team is greatly increased.

For all the above reasons, coordination techniques have been applied to multiple application domains such as wireless sensor networks (Rogers et al., 2009), teams of unmanned vehicles deployed in disaster response scenarios (Scerri et al., 2005) and scheduling multiprocessor jobs in distributed computational environments (Sultanik et al., 2007). Among these, the domain of robotics for disaster response appears as a very challenging domain for testing coordination mechanisms, because of its complexity, dynamism and unpredictability (Bethke et al., 2008). Within such settings, teams of autonomous robotic agents such as unmanned aerial or ground vehicles (UAVs and UGVs respectively, see



(a) A team of autonomous UAVs (Courtesy of the Penn State University)

(b) A team of autonomous UGVs (Courtesy of the University of Michigan)

FIGURE 1.1: Two types of unmanned vehicles.

Figure 1.1 for examples of such vehicles) are often deployed to achieve accurate situational awareness—the ability of a robotic agent to make sense of, and predict, what is happening in an environment—at the scene of a disaster. Examples include measuring the temperature of a building on fire, searching for a drifting life-raft after the sinking of a ship, or collecting imagery from the most dangerous areas of the scene of a disaster. By adopting a coordinated response, the agents are able to maximise the amount of collected information since they will avoid making redundant decisions, and thus, they will avoid exploring the same area more than once and they will avoid making sub-optimal decisions, such as exploring areas which have little importance for the operation.

Now, a variety of mechanisms have been developed to address coordination problems (see Chapter 2 for more details). Typically, decentralised algorithms are advocated for these settings since they are scalable and robust to the failure of one or more agents

(Bethke et al., 2008; How et al., 2009). These approaches have been studied in research in both robotics and multi-agent systems. To date, research on robotics has principally focused on developing and evaluating a general unmanned architecture to coordinate the decisions of the team of agents. Within this architecture, coordination algorithms are often defined as simple distributed protocols that allow the agents to negotiate over a collective decision in real time and whilst operating within an environment (Bethke et al., 2008; Grocholsky, 2002). However, whereas they have been shown to work effectively when deployed on real UAVs or UGVs, these algorithms still lack a systematic study, that establishes their qualities and, most importantly, that focuses on improving them. In contrast, research in artificial intelligence, and, most importantly, research in multi-agent systems has focused on developing more sophisticated coordination algorithms allowing the agents to make optimal or bounded approximate decisions despite incurring a higher cost in terms of computation and communication (Modi et al., 2005; Rogers et al., 2011). However, to date, these algorithms have only been studied in simulation, and their effectiveness when deployed in the real world still remains an open issue.

Most importantly, the former algorithms, both the ones studied in robotics and the ones in multi-agent systems, work by assuming that the quality of the agents decisions can be measured using real values. In other words, a real valued function is defined over the agents' decisions, that assigns higher values to those decisions that improve the team performance. This is the case in the domain of robotics for disaster response, in which multiple examples of such functions exist, such as the mutual information gain, the entropy and the cumulative probability of detection (Grocholsky, 2002; Cole, 2009).

However, not all decisions can be valued using a real valued function. In fact, there exists multiple real world settings in which agents make complex decisions whose outcome cannot be determined by such function. The reason for this is that a real function outputs values that are totally ordered (i.e. they are fully comparable and a single optimum exists). Unfortunately, making decisions in the real world is often a complex process which might require trading off different factors and parameters. This process cannot simply be encapsulated within a real value function. Hence, new functions need to be defined, which measure the quality of the agents' decisions using partially rather than totally ordered values. In so doing, the agents decisions might not be fully comparable and actually, multiple equivalent decisions might exist. However, using these functions will incorporate the complexity of the real world in the agents' decision making process.

Now, there exists numerous problems involving partially ordered functions to model complex decisions. However, in most cases, literature does not provide algorithms capable of addressing such problems. A first example is the use of multi-objective functions to measure complex agents' decisions. Focusing on disaster response, consider a setting in which a number of UAVs are deployed to search for drifting life-rafts after the sinking of a ship. In reality, the aim of their mission is not only to find all the life-rafts

in the least amount of time, but to report their positions with accurate precision to the first responders so that they can take the victims to safety. To do so, the UAVs need to track the life-rafts to localise their position with precision. However, these two objectives conflict, since the optimisation of one of them is likely to result in the detrimental performance of the other (e.g. if a UAV is searching it cannot track a discovered life-raft). As a consequence, they are partially ordered, since they cannot be optimised independently.

A second example involves defining partially ordered functions to measure the quality of the agents' decisions when their outcome is uncertain. This is what happens in many real world settings, particularly in disaster response (see Chapter 2 for details) in which the deployed agents are likely to have noisy sensors and, as a consequence, their position and observations are uncertain. In these settings, the value of the scalar function corresponding to each possible outcome of the agents' decisions is uncertain. Hence, the agents do not know with precision what the outcome of their decisions is going to be. To face this uncertainty then, the agents have to decide on a collective attitude to face the risk associated with making decisions that might result in poor outcomes. In fact, there exists an extensive literature on decision making under uncertainty that indicates that preferences over outcomes should be determined by a risk profile encoding the agents' willingness to expose themselves to uncertain outcomes in prospect of a higher reward (Levy, 2006). To do so, it becomes necessary to trade off between the uncertainty of the agents' decisions and the potential value that these decisions can yield which result again in a partial order. For instance, consider a team of UAVs deployed to search for an unknown number of drifting life-rafts. In this setting, the agents' decisions are typically defined as the areas to explore and their outcomes depends on the number of discovered life-rafts. Naturally, this number is unknown, therefore the expected value is typically used. In light of this, the agents might prefer exploring areas with a certain rather than uncertain number of casualties, despite possibility that the certain number may be lower than the uncertain one. This type of attitude is typically defined as a risk averse behaviour. Conversely, they might prefer exploring areas with an uncertain number of casualties, hoping to discover a large number of casualties (i.e. a very good outcome) despite the risk of ending up in an empty area (i.e. a very poor outcome). This type of attitude is typically defined as a risk seeking behaviour. Finally, they might be completely neutral to the uncertainty associated with the outcome of their decision and simply decide which areas to explore next considering the areas' expected number of life-rafts.

In addition, within other application domains the agents' decisions might be even more complex. Indeed, there are many settings in which no *a priori* information might be known about the environment where the agents are deployed. Moreover, the uncertainty characterising the outcomes of the agents' decisions might be unknown or impossible to

define. In such cases, it is then impossible to define a function to determine the quality of the agents' decisions. For instance, in the example discussed above, nothing might be known about the positions of the life-rafts before the UAVs are actually deployed. Hence no estimated or expected number of casualties can be defined. In this setting, the UAVs do not know what the actual outcome of their decisions is going to be. For instance, they might assume that a certain area contains a high number of life-rafts, and thus, that exploring it might be highly rewarding. However, in reality the area might contain a limited number of life-rafts and consequently, it is not the best area to explore. Hence the estimate of these outcomes should be refined, or learnt during the operation. Such estimates, however, will be defined over multiple inter-dependent parameters for which a trade off might again be necessary. Hence, they are again partially ordered.

Now, some literature has tried to address the three example of complex decisions presented above. For instance, various algorithms have been proposed to address multi-objective problems. However, they are either centralised or unable to run in real time due to their complexity (see Chapter 2 for more details). In addition, a vast literature exists in incorporating risk in decision making under uncertainty. However doing so for coordination problems involving multiple agents remains an open challenge since current solution techniques can address uncertain outcomes such as those presented above, but fall short in taking into account the risk that each of these outcomes might incur (Atlas and Decker, 2010; Léauté and Faltings, 2011). Finally, whereas a variety of literature exists about online learning algorithms, these are either single agent approaches, or approximate, and thus, likely to result in a poor performance (Lai and Robbins, 1985; Léauté and Faltings, 2009).

To address these shortcomings, the theory of coordination algorithms (i.e. the design of the algorithms and the study of their properties) and their practice (i.e. the analysis of their performance either in simulated or real world environments) has become a key research challenge. The aim of practical research is to close the gap between existing coordination algorithms and the real world. Indeed, as discussed above, whereas numerous algorithms have been defined, their effectiveness to coordinate robotic agents deployed in the real world still remains an open question. Investigating such effectiveness requires implementing these algorithms within a real coordination system and verifying their performance on real robotic agents such as UAVs or UGVs, considering all their requirements (e.g. robustness and scalability) and limitations (e.g. battery capacity and computational power). In contrast, the aim of theoretical research is to study coordination problems from an abstract perspective. The key focus is to derive new algorithms that enrich the decision making capacities of the agents and thus allow them to make complex decisions involving multiple inter-dependent parameters. Indeed, as discussed above, current coordination algorithms are not yet capable of handling problems involving multiple objectives, uncertainty or a complete lack of information. Deriving

these new algorithms, then, requires understanding their features and analysing their properties in a simulated environment before actually testing them in the real world.

Against this background, the key purpose of this thesis is to address the above mentioned research challenges. In the remainder of this chapter, we will describe, in more detail, the aims of this work before detailing our contributions.

1.1 Research Objectives

The objectives of this thesis are twofold. The first is to design new coordination algorithms to enhance the decision making capacities of a multitude of agents cooperating to solve a problem. In so doing, we aim to start reducing the gap between the decision making capabilities provided by the current state-of-the-art coordination algorithms and the capabilities that are actually necessary to solve real world problems. The second objective is to understand the requirements that are necessary to coordinate robotic agents in the real world by deploying state-of-the-art coordination algorithms to address real world coordination problems.

In more detail, coordination algorithms need to satisfy a set of specific requirements to be successfully deployed to solve real world problems. These requirements arise when a potentially large number (i.e. tens) of agents are deployed, which will be sensing, acting and making decisions in a complex and potentially unknown environment. To identify these requirements, a scenario is proposed, taken, again, from the domain of robotics for disaster response. In fact, this thesis will focus principally on this domain to illustrate the most relevant concepts. Whereas it is well acknowledged that there are multiple domains in which coordination technologies can be applied (see the beginning of this chapter for some references), robotics stands out as one of the most challenging. This is due to a number of factors such as the uncertainty which is endemic in most robotic domains, the complexity of building effective robotic systems and, most importantly, the impact that any effective robotic system can have in the real world. As a consequence, robotics for disaster response, and particularly for situational awareness, will constitute the main application domain of this thesis. The scenario, which is inspired by Haiti's 2010 earthquake, proceeds as follows:

Scenario. *Haiti, 2010. A catastrophic magnitude* 7.0 M_w *earthquake has struck the country's capital Port au Prince. Over* 3 000 000 *people living within the area are affected by the earthquake and numerous buildings have collapsed or have been severely damaged.*

The principal objective of the rescue forces is to save as many casualties as possible. To achieve this, rescue forces need to accurately assess the situation at the scene of the

disaster. Naturally, this implies collecting accurate information about the most critical areas of the capital, such as collapsed or damaged buildings. However, not only is the number of areas to explore extremely large, but most of these also present hazards to first responders, who are already fully occupied providing assistance to the casualties who have already been found. Fortunately, the first responders have been provided with a fleet of thirty UAVs and UGVs that can be deployed to collect information about these critical areas.

In more detail, each first responder is provided with a personal digital assistant (PDA) that he uses to submit imagery collection tasks to the vehicles. UAVs are then deployed to collect aerial imagery of dangerous areas such as building on fire or of large areas where buildings have collapsed. Similarly, UGVs are deployed to collect imagery of indoor environments, such as damaged buildings, so as to understand if casualties are still in there. Both these vehicle types are then equipped with the latest state-of-the-art agent-based technology for coordinated decision making, that allows them to operate autonomously over the area of the disaster and report all the gathered information back to the first responders.

During the operation, the vehicles face a variety of challenges. Some vehicles fail, while new ones are deployed since the areas and the number of buildings to explore are too large. All vehicles have limited battery capacity and limited computation and communication resources. Their sensors (e.g. their camera and GPS locator) are noisy and imprecise. Moreover, after the earthquake, most communication went down, and thus the position of most of the casualties in Port au Prince is unknown prior to the agents deployment. Nonetheless, due to their robust and flexible decision making capabilities, the vehicles are capable of reporting enough information to build an accurate picture of the disaster area, containing the location of most of the casualties.

Now, the scenario allows us to identify a more detailed set of requirements to guide the design of efficient coordination algorithms. These are defined as follows:

- **Generality**: This requirement demands coordination algorithms to be applicable across a wide range of different settings. For instance, as described by the scenario, the deployed agents face two different challenges: the first consists of providing aerial imagery and requires that the UAVs can auto-determine which areas need to be explored and determine how much imagery needs to be collected for each of them; the second consists of exploring the interior of buildings to build a map providing first responders with the position of all the discovered casualties. Hence, this requirement deals with the definition of coordination techniques that are re-usable and, that, as a consequence, can be applied to other scenarios out of the scope of this work.

- **Performance Guarantee**: This requirement demands that coordination algorithms provide theoretical guarantees on the performance that they yield. For instance, the scenario describes a critical setting in which a team of UAVs are deployed to collect aerial imagery of dangerous areas such as collapsed buildings or buildings on fire. Within this setting, first responders will be more amenable to delegating imagery collection tasks to the robotic agents if they know that the coordination mechanism allows the agents to perform either optimally or with a guaranteed bound to the optimal behaviour. More generally, this requirement deals with the definition of theoretical proofs, that provide developers with a certificate of the algorithm's quality and thus help them to choose which coordination technology to deploy.

- **Robustness**: This requirement demands coordination algorithms to allow the agents to continue the operation whenever communication is limited or if the failure of one or more agents occurs. For instance, in the setting described in the scenario, some agents may leave the operation due to component failure. Moreover, since radio communication is typically severely limited within damaged buildings, the UGVs will likely have trouble communicating during their operation. In both cases the other agents should be able to continue the operation without being significantly affected. Hence this requirement deals with the definition of coordination techniques whose performance is not going to be unduly affected by technical constraints, such as the failure of one agent or its limited communication capacities.

- **Scalability**: This requirement demands coordination algorithms to yield efficient performance in cases where large number of agents need to be deployed[1]. For instance, in the setting described by the scenario, the number of deployed agents is increased to complete the operation. Nonetheless, the performance of the entire team should remain unaffected. Hence, this requirement deals with the definition of coordination techniques that are capable to handle the interactions with a growing number of neighbouring agents.

- **Resource-awareness**: This requirement demands coordination algorithms that allow agents to make high quality decisions given their limited computation and communication capacities. For instance, the scenario describes a setting in which the deployed vehicles have limited battery capacity and computational power. Nonetheless this should not unduly affect their decision making capability. Hence, this requirement deals with the definition of a coordination technology that allows robotic agents to operate efficiently in real-time operations.

- **Complex Interactions**: This requirement demands coordination algorithms to allow the agents to make complex decisions involving trading off between various

[1]The algorithms presented in this thesis will be tested over tens of agents. Scaling to a higher number might be possible as well, but is out of the scope of this work

parameters of the problem. As we have seen, the agents might be making complex decisions involving multiple and conflicting objectives. Consider the scenario described above, where UAVs are deployed to complete imagery collection tasks. Their objective is to maximise the number of completed tasks. However, some of the areas where imagery needs to be collected might contain hazards for the agents, such as fire, water or obstacles. Hence, to prevent the loss of the vehicles, the agents might consider minimising the number of completed dangerous tasks as a second objective. However, the optimisation of one objective may result in detrimental performance in terms of the other (i.e. agents will not complete dangerous tasks, and thus the number of completed tasks is likely to decrease). As a second example, the agents might be making complex decisions whose outcome is uncertain. In this setting, the agents might risk making decisions which might yield a poor outcome. For example, in the scenario the agents' sensors are noisy. This means that each agent's measurements and position is uncertain. By adopting a collective risk profile (neutral, averse or seeking), the agents then become capable of trading off the reward associated with each uncertain decision (i.e. the number of casualties they are expected to find). To do so, however, it becomes necessary to trade off between the expected outcome of each decision and their uncertainty. As a final example, the agent might be making complex decisions whose outcome need to be learnt during the operation. Again, the scenario describes the environment where the agents are deployed as unknown, uncertain and unpredictable. Moreover, these agents have noisy sensors. Thus, the outcome of their decisions cannot be defined *a priori* due to the level of uncertainty and the lack of knowledge. These outcomes then need to be learnt during the operation, by considering not only their estimated value, but, most importantly the bound on the uncertainty defining their estimate. More generally, this requirement deals with the definition of coordination technologies that can enhance the decision making capabilities of a multi-agent system, by making the agents capable of handling complex interactions.

To address these requirements, *decentralised* coordination techniques have been advocated, since they yield scalable solutions, exhibit rapid response times by exploiting the locality of the agents' interactions and are made robust to failure by avoiding the existence of a centralised point of control (Jennings, 2001; How et al., 2009). A variety of these coordination algorithms have then been proposed within the robotics (Bethke et al., 2008; How et al., 2009), artificial intelligence (Modi et al., 2005; Rogers et al., 2011) and machine learning (Guestrin et al., 2003; Auer et al., 2002) communities. Among these, the use of algorithms that solve distributed constraint optimisation problems (DCOPs) have been advocated, since they position themselves well with respect to the robustness, scalability and rapidity requirements mentioned above (Modi et al., 2005). In particular, the DCOP framework is a well known and studied model that represents coordination

problems as constraint optimisation problems. In a DCOP, variables represent the decisions that the agents can make, while constraint functions represent their interactions (see Chapter 2 for a more thorough description). In more detail, each agent controls a variable representing its decisions, and the agent's objective is then to maximise a global objective function defined as the sum of all the local constraint functions representing the interactions between the agents. Due to this generality, DCOPs have been shown to be a very effective framework to represent coordination problems (Modi et al., 2005). For this reason, the DCOP framework is the foundation over which we will build the algorithms described in this thesis (other methods are briefly discussed and analysed in Chapter 2).

In addition, one particular DCOP algorithm, max-sum, has been shown to position itself particularly well with respect to the requirements defined above (Stranders, 2010). The principal reason is because it builds upon the Generalised Distributive Law (GDL) (Aji and McEliece, 2000), a well known framework that exploits the factorisability of a number of optimisation problems (including DCOPs) and can thus provide effective solutions whilst preserving the agents' limited computational and communication resources. In addition, the GDL framework can be used to transform max-sum into either optimal (DPOP and Action GDL (Petcu and Faltings, 2005; Vinyals et al., 2011)) or bounded approximate (Rogers et al., 2011) algorithms. Naturally, computing bounded approximate or optimal solutions requires more computation and communication, which limits, at the moment, the practical applicability of the former algorithms. However, they still constitute a very good starting point for the theoretical purposes of this work. For these reasons, max-sum is the key algorithm which we are going to use in this thesis.

However, despite its potential, max-sum has not yet been deployed on a real system. It has only been tested in simulation, which lacks the dynamism and the heterogeneity of the real world. Hence, max-sum's robustness and flexibility for real applications has not been tested in practice. Moreover, max-sum's performance depends greatly on the way it is applied to a problem. In fact, this performance is affected by both the way in which a generic problem is encoded to form the input of the algorithm, and the way the algorithm is decentralised between the available sources of computation. However, whereas a variety of ways to apply the algorithm have been described in the literature, a general framework that discusses and analyses these issues is absent. Most significantly, for the work presented here, GDL algorithms (comprising max-sum) are not yet capable of addressing the requirement of complex interactions. In particular, whereas, various frameworks and algorithms have been defined to address complex interactions representing either uncertainty, online learning and multiple objectives (Atlas and Decker, 2010; Taylor et al., 2010; Fitzpatrick and Meertens, 2003; Maheswaran et al., 2005), thus far, no framework nor solution technique exists that addresses the general problem of representing decisions over multiple inter-dependent parameters.

The focus of this thesis is, therefore, to address these shortcomings. In the next section, we describe our contributions towards this end.

1.2 Research Contributions

The contributions of this thesis can be divided into two main categories: practical and theoretical. The former focus on evaluating the performance of the max-sum algorithm when deployed in the real world. In more detail, we present a study of the deployment of the max-sum algorithm. First, we introduce a methodology that the less-experienced developer can use to deploy max-sum for problems related to situational awareness. Second, we present a case study whereby we apply our methodology to deploy max-sum to coordinate a team of unmanned aerial vehicles to provide live aerial imagery to first responders operating in the area of a disaster.

The theoretical contributions, which constitute the main thrust of this thesis, then focus on deriving new GDL algorithms to address the requirement of complex interactions. To do so, we generalise the canonical DCOP framework and derive a new class of problems, namely partially ordered constraint optimisation problems (PO-DCOPs) which define DCOP's constraint functions as partially ordered functions. In particular, the notion of partial order is introduced to encompass all the settings in which the agents' joint decisions cannot be measured using canonical scalar functions (which are by definition totally ordered). In addition, partially ordered instances might not be comparable anymore. This implies that a single maximum might not exist anymore. Rather, a set of multiple *non-dominated* alternatives might co-exist, which might not be comparable between each other and which might all need to be considered as potential solutions.

Given these observations, solving these problems requires deriving new algorithms capable of handling the above mentioned partial order. To do so, as we show in this thesis, a very good starting point is to exploit the GDL framework. Indeed, as described in Section 1.1, the GDL exploits some structural property of DCOPs (i.e. the fact that they are defined over a *commutative semi-ring* as we will see in Chapter 2) to solve them in an efficient manner. This property can be used to define new algorithms that can solve our PO-DCOPs. Moreover, we can use the same transformations used for canonical DCOPs to derive optimal and bounded approximate algorithms equivalent to DPOP, Action GDL and bounded max-sum. Using the PO-DCOP framework, we can then define three new classes of DCOPs, namely multi-objective DCOPs (MO-DCOPs), risk aware DCOPs (RA-DCOPs) and multi arm bandit DCOPs (MAB-DCOPs) each representing one of the complex interactions described in Section 1.1 and derive new algorithms based on the GDL to solve them. Indeed, we show how each of the former

Requirement	Max-sum	Bounded Max-sum	Action-GDL	DPOP
Generality	+	+	+	+
Performance Guarantee	-	+	+	+
Robustness	+	+	+	+
Scalability	+	+	+	+
Resource-awareness	+	+	-	-
Complex Interactions	-	-	-	-

TABLE 1.1: The list of requirements satisfied by the GDL algorithms. In the table "+" stands for satisfied and "-" for not satisfied.

classes cannot be solved by traditional DCOP algorithms, since a partial order exists between each problem's solutions.

In more detail, these contributions are described in the following chapters:

Chapter 3: Application of Coordination for Disaster Response

This chapter presents a study of the deployment of the max-sum algorithm within a system to coordinate a team of UAVs deployed over the scene of a disaster, to provide the first responders at the scene with live aerial imagery of the most critical areas. Each first responder is provided with a personal digital assistant (PDA) that he can use to submit imagery collection tasks. The coordination system works by dynamically casting the corresponding task allocation problem into a DCOP in which the local constraint functions represent utility functions measuring the allocation of one specific task to one or more of the UAVs that can attend it. The first contribution of this chapter is then a novel utility defined to carefully weight the problem's constraints, such as a task's importance and the UAV's battery capacity, to maximise performance. The UAVs then coordinate to maximise the number of completed tasks with the highest importance (i.e to maximise the sum of the tasks' utilities).

The remaining contributions of this chapter are then twofold. First, a methodology is proposed that a developer can use to deploy the max-sum algorithm to coordinate robotic agents to achieve accurate situational awareness in disaster scenarios. In so doing, a set of general rules is identified, that unifies the different ways in which the algorithm can be applied to problems related to situational awareness. Second, a case study is presented whereby the methodology is used to deploy the coordination system described. This system is evaluated in both simulation and over two real UAVs.

Simulations are used to ascertain the advantage of using the new task's utility. Indeed, since it is composed of multiple parameters it could impose some severe constraints on the coordination algorithm and, as a consequence, yield poor performance in terms of the quality and the number of completed tasks. To verify this hypothesis, we benchmark the former utility against more standard less constrained utilities to evaluate its performance. Our results show that it yields a better trade off between the quantity and quality of

completed tasks. Second, the system is deployed on two real UAVs and the effectiveness of max-sum is evaluated in different scenarios, thus showing its practical viability.

To summarise, this chapter makes the following contributions to the state of the art:

- We develop a methodology to apply max-sum to problems related to situational awareness. We unify the various ways of using max-sum that exist in literature and organise them into a sequence of steps that can guide a developer to the successful deployment of the algorithm.
- We present a coordination system for disaster management, which could be potentially deployed for real operations.
- We introduce a novel utility function in which several constraints are carefully weighted to maximise performance. We then empirically evaluate our utility against similar ones but which do not take all the constraints into account and we show that our utility yields a better trade off between the quantity and quality of completed tasks.
- We evaluate the system by deploying it on two hexacopter UAVs in three different settings and ascertain its practical viability.

The research presented in this chapter led to the following papers:

- Delle Fave, F. M., Rogers, A., Xu, Z., Sukkarieh, S. and Jennings, N.R. (2012) Deploying the Max-Sum Algorithm for Coordination and Task Allocation of Unmanned Aerial Vehicles for Live Aerial Imagery Collection. In: *Proceedings of the IEEE International Conference on Robotics and Automation* (ICRA). St. Paul, USA. pp. 469–476
- Delle Fave F. M., Farinelli A., Rogers, A. and Jennings N. R. (2012) A methodology for deploying the Max-Sum Algorithm and a Case Study on Unmanned Aerial Vehicles. *Proceedings of the Twenty Fourth Conference for Innovative Application for Artificial Intelligence* (IAAI 2012). Toronto, Canada. In press.
- Delle Fave F. M., Rogers, A. and Jennings N. R. (2012) ARGUS: A Coordination System to Provide First Responders with Live Aerial Imagery of the Scene of a Disaster (Demonstration). In: *Proceedings of the Eleventh Conference on Autonomous Agents and Multi-Agent Systems* (AAMAS 2012). Valencia, Spain. pp. 345–346.

Chapter 4: A Class of Coordination Algorithms for Partially Ordered Coordination Problems

Requirement	B-MOMS
Generality	+
Performance Guarantee	+
Robustness	+
Scalability	+
Resource-awareness	+
Complex Interactions	+

TABLE 1.2: The list of requirements satisfied by the B-MOMS algorithm. In the table "+" stands for satisfied and "-" for not satisfied.

This chapter presents a general framework to represent settings in which agents are required to make complex joint decisions which cannot be measured using the real-valued constraint functions used in traditional DCOPs.

In more detail, the contributions of this chapter are twofold. First, we propose a novel framework, namely partially ordered distributed constraint optimisation problems (PO-DCOPs) to model coordination problems in which the agents are making complex decisions resulting in a partial order. A PO-DCOP generalises a DCOP by defining the constraint functions as partially rather than totally ordered functions. Such functions are typically represented as vector functions containing the different parameters defining the partial order. Examples include multiple objectives (see Chapter 5), mean and variance (Chapter 6) and mean and upper confidence bound (Chapter 7). In so doing, we propose a novel class of optimisation problems that allow us to model complex decisions and, as a consequence, allows us to encompass more realistic coordination problems such as those presented at the beginning of this chapter. Second, we propose the partially ordered generalised distributive law (PO-GDL) a class of algorithms which builds upon the generalised distributive law (GDL) to solve PO-DCOPs. In more detail, we exploit the well known flexibility of the GDL to derive efficient optimisation algorithms, such as the dynamic programming optimisation protocol (DPOP), Action GDL and bounded max-sum to design algorithms to solve PO-DCOPs. Three instantiations of the PO-GDL will then be used in the following chapters, to solve DCOP involving complex interactions including multiple objectives, uncertainty about the outcome of the agents decisions and online learning.

The research presented in this chapter led to the preparation of the following journal paper:

- Delle Fave, F. M., Stranders, R., Rogers, A. and Jennings, N. R. Partially Ordered Distributed Constraint Optimisation Problems and Algorithms. To be submitted.

Chapter 5: A Class of Algorithms for Partially Ordered Coordination Problems involving Multiple Objectives

This chapter introduces a class of algorithms able to address coordination problems where the agents' decisions are characterised by multiple conflicting objectives. To achieve this, we first define multi-objective DCOPs (MO-DCOPs) an instantiation of DCOPs where local constraint functions are defined over multiple objectives and, as a consequence, where the solutions of the corresponding global function are partially ordered. To define these solutions, we then exploit the well known concept of Pareto optimality (Pareto, 1906), that allows us to discriminate between partially ordered alternatives defined over multiple objectives. In more detail, we use Pareto optimality to derive a class of algorithms to solve MO-DCOPs. In so doing, we propose PDS-GDL, a new class of algorithms for solving MO-DCOPs that exploits the GDL. This class is based on an algebraic structure, the Pareto-dominance / sum (PDS) semi-ring, that can be used to transform any GDL-based algorithm into an algorithm for solving MO-DCOPs.

To solve MO-DCOPs optimally, these algorithms have to keep track of multiple Pareto optimal partial solutions. Since the number of solutions can be large, we instantiate NDS-GDL to derive an equivalent of the bounded max-sum algorithm, namely the bounded multi-objective max-sum algorithm (B-MOMS), to solve MO-DCOPs. We then evaluate how the increase in the number of Pareto optimal solutions affects the performance of the algorithm. In so doing, we present an extensive empirical evaluation of B-MOMS by benchmarking against a centralised optimal algorithm on a multi-objective extension of the graph colouring problem, a standard test problem for DCOP algorithms (Modi et al., 2005). We show that the approximation ratio never exceeds 2 (i.e. the solution's quality is never worse than 50% from the optimal), even for extremely constrained problems (i.e. fully connected graphs), and is less than 1.5 for graphs where constraints exist between 20% of all pairs of agents for 14 variables. In terms of the requirements defined in Section 1.1, it should be noted that the complexity of the problem is increased since the algorithm now has to keep track of multiple Pareto optimal solutions. Thus, it cannot satisfy the requirements of generality, robustness, scalability and resource-awareness, to the same extent as it does in the single-objective case. Nonetheless, the results indicate that the runtime required by B-MOMS never exceeds 30 minutes, even for maximally constrained graphs with 100 agents, positioning it well within the confines of many real-life applications. Hence, the empirical evaluation shows that the algorithm is able to provide performance guarantees whilst being resource aware even in extremely complex problems (maximally constrained graphs). Table 1.2 then summarises the way in which the algorithm positions itself with respect to the requirements defined in Section 1.2.

Thus, this chapter makes the following contributions to the state of the art:

- We formally describe MO-DCOPs, an instantiation of PO-DCOPs in which the

Requirement	Action NDC-GDL (s)	Action NDC-GDL (n)
Generality	+	+
Performance Guarantee	+	-
Robustness	+	+
Scalability	+	+
Resource-awareness	-	+
Complex Interactions	+	+

TABLE 1.3: The list of requirements satisfied by the Action NDC-GDL algorithm with both a sufficient (s) and a necessary condition (n). In the table "+" stands for satisfied and "-" for not satisfied.

local constraint functions and the global utility function are characterised by multiple conflicting objectives.

- We define the Pareto-Dominance/Sum (PDS) semi-ring which can be used to transform any GDL-based algorithm into a new algorithm for solving MO-DCOPs. In so doing, we theoretically demonstrate the existence of a new class of algorithms (the NDS-GDL) that exploits the GDL for solving MO-DCOPs.
- We instantiate the PDS-GDL to produce the B-MOMS algorithm, a novel generalisation of bounded max-sum, which we empirically demonstrate to be able to provide performance guarantees on the solution recovered whilst preserving the agents' limited resources, despite the increased complexity of the problem.

The research presented in this chapter led to the publication of the following paper:

- Delle Fave, F. M., Stranders, R., Rogers, A. and Jennings, N. R. (2011) Bounded Decentralised Coordination over Multiple Objectives. In: Proceedings of the Tenth International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2011). Taipei, Taiwan. pp. 371–378.

Chapter 6: A Class of Algorithms for Partially Ordered Coordination Problems involving Risk Awareness

This chapter introduces a class of algorithms able to address coordination problems where the outcome of the agents' decisions is uncertain and thus, where the agents are required to make decisions while carefully weighting the risk of exposure to poor outcomes. To achieve this, we first introduce risk-aware distributed constraint optimisation problems (RA-DCOPs), a specialisation of PO-DCOPs that represents the outcomes of the constraint functions as random variables, distributed according to a given probability density function (pdf), and, most importantly, where the global objective is to maximise the expectation of a utility function representing the agents' collective risk-profile. We then derive NDC-GDL, a new class of algorithms for solving RA-DCOPs that exploits

the PO-GDL. To do this, we define the Non-Dominance/Convolution (NDC) semi-ring that can be used to transform any GDL-based algorithm into an algorithm for solving RA-DCOPs. To solve RA-DCOPs optimally, these algorithms have to keep track of multiple non-dominated partially ordered solutions. Given this potentially large number, we then show that different types of domination conditions can be used to select these solutions. Optimal conditions propagate the smallest number of pdfs without losing the guarantee of optimality, but do not always have a closed form expression. Alternatively, sufficient (which achieve optimality at the cost of a higher overhead), and necessary (which incur lower costs but can sacrifice optimality) conditions can be used. Hence, choosing one condition instead of another affects the way in which NDC-GDL algorithms satisfy the requirements defined in Section 1.2. In particular, sufficient conditions provide quality guarantees (optimality) whilst utilising more resources, whereas necessary conditions sacrifice performance guarantees, but incur lower costs. In light of this, we instantiate the NDC-GDL to produce an equivalent of the Action GDL algorithm (namely Action NDC-GDL) which we instantiate with an optimal, a sufficient and a necessary condition. In so doing, we empirically demonstrate how the use of a necessary condition, which is equivalent to the use of the canonical Action GDL algorithm to solve an RA-DCOP, can lead to significantly suboptimal performance, especially as the level of risk aversion is increased. This is summarised in Table 1.3.

Thus, this chapter makes the following contributions to the state of the art:

- We formally describe RA-DCOPs, a specialisation of PO-DCOPs in which the outcome of each constraint function is a random variable.
- We define the Non-Dominance/Convolution (NDC) semi-ring which exploits the PO-GDL to solve RA-DCOPs. In so doing, we theoretically demonstrate the existence of a new class of algorithms (the NDC-GDL) that can be used to instantiate any GDL algorithm to solve RA-DCOPs.
- We theoretically demonstrate the existence of different dominance conditions for NDC-GDL algorithms, namely optimal, sufficient and necessary conditions. These can be used when pdfs do not have a closed form expression and to trade off between the solutions' quality and the computational cost.
- We empirically demonstrate the need for the NDC-GDL class to solve RA-DCOPs by showing that transforming them into DCOPs can lead to significantly suboptimal performance, especially as the level of risk aversion is increased.

The research presented in this chapter led to the publication of the following paper:

Stranders, R., Delle Fave, F. M., Rogers A., and Jennings, N. R. U-GDL: A decentralised algorithm for DCOPs with Uncertainty. In: *Proceedings of the AAMAS workshop on Optimisation for Multi-Agent Systems* (OptMAS 2011). Taiwan, Taipei.

Chapter 7: A Class of Algorithms for Partially Ordered Coordination Problems involving Online Learning

This chapter introduces a class of algorithms able to address coordination problems where the outcome of the agents' decisions is uncertain and impossible to determine before the agents' deployment. In fact, within this setting, the agents are no longer faced with a one shot optimisation problem as encoded in a canonical DCOP. Rather, we now have a problem in which agents have to coordinate to solve a sequence of "DCOP-like" problems in order to simultaneously reduce uncertainty about the local constraint functions (*exploration*) and maximise the global utility (*exploitation*). This implies the need for striking a balance between exploration and exploitation. Focusing solely on exploration results in certainty about the agents' environment, but wastes resources by taking suboptimal actions. Similarly, consistently taking the joint action that is currently believed to be the best is also suboptimal because this belief might be incorrect.

The multi arm bandit (MAB) community has addressed the trade off between exploration and exploitation in a principled fashion from a single agent perspective (Lai and Robbins, 1985; Auer et al., 2002; Vermorel and Mohri, 2005). A MAB is a simple analytical tool which models decision making under uncertainty. In more detail, a MAB models a slot machine with multiple arms, each of which yields a reward, drawn from an unknown but fixed probability distribution. The aim of the problem is to sequentially pull the arms so as to maximise the cumulative reward over a finite time horizon.

Thus, to address the challenge of coordinating when the outcome of the agents' decisions is uncertain and unknown, we propose in this chapter a class of algorithms that combine the robustness and scalability of GDL algorithms with the optimal exploration/exploitation trade off that the MAB algorithms provide. In more detail, we first introduce MAB-DCOPs, a specialisation of PO-DCOPs, in which each local utility function becomes a MAB. This effectively models both the stochastic nature of realistic decentralised coordination problems, as well as the absence of *a priori* knowledge. Unlike a DCOP, a MAB-DCOP is not a single shot optimisation problem, but rather a sequential problem in which agents need to coordinate their joint actions over multiple time steps, so as to maximise the cumulative global utility received over a finite time horizon. We then derive the upper confidence bound GDL (UCB-GDL), a new class of algorithms for solving MAB-DCOPs that exploits the GDL and generalises the well known upper confidence bound (UCB) technique for MAB to the multi-agent setting. As was done in the previous chapters, we define these algorithms by formalising the UCB-GDL semi-ring that can be used to transform any GDL-based algorithm, such as DPOP, max-sum, or Action-GDL, into an algorithm for solving MAB-DCOPs. We then empirically evaluate one of these algorithms, which we refer to as Heist, in a reproducible controlled environment, and show that it outperforms other state of the art techniques from the MAB and DCOP literature (among which max-sum and ε-first) by

Requirement	HEIST
Generality	+
Performance Guarantee	+
Robustness	+
Scalability	+
Resource-awareness	-
Complex Interactions	+

TABLE 1.4: The list of requirements satisfied by the HEIST algorithm. In the table "+" stands for satisfied and "-" for not satisfied.

up to 1.5 orders of magnitude on MAB-DCOPs. In so doing, we show that UCB-GDL algorithms are resource aware and provide performance guarantees on the regret of the global cumulative performance as described more precisely by Table 1.4.

Thus, this chapter makes the following contributions to the state of the art:

- We introduce MAB-DCOPs, a specialisation of PO-DCOPs, to represent decentralised coordination problems with uncertainty and the absence of *a priori* knowledge about local utility functions.
- We define the UCB-GDL semi-ring which can be used to transform any GDL-based algorithm, such as DPOP, max-sum, or Action-GDL, into an algorithm for solving MAB-DCOPs. In so doing, we theoretically demonstrate the existence of a new class of algorithms (the UCB-GDL) that exploits the GDL for solving MAB-DCOPs.
- We show that any UCB-GDL algorithm can provide optimal asymptotic bounds on the regret of the global cumulative utility.
- We use the UCB-GDL to derive a version of the Action GDL algorithm to solve MAB-DCOPs and demonstrate its effectiveness in a reproducible controlled environment.

The research presented in this chapter led to the publication of the following paper:

- Stranders, R., Tran-Thanh, L., Delle Fave, F. M., Rogers, A. and Jennings, N. R. (2012) DCOPs and bandits: Exploration and exploitation in decentralised coordination. In: *Proceedings of the Eleventh International Conference on Autonomous Agents and Multi-Agent Systems* (AAMAS 2012). Valencia, Spain. pp. 289–297.

1.3 Thesis Structure

This thesis is organised as follows:

Chapter 2 discusses related work.

Chapter 3 introduces our study on the real world deployment of max-sum on the case study of disaster response.

Chapter 4 introduces PO-DCOPs and describes the PO-GDL methodology to derive algorithms to solve them.

Chapter 5 introduces MO-DCOPs and describes the NDS-GDL class of algorithms to solve them.

Chapter 6 introduces RA-DCOPs and describes the NDC-GDL class of algorithms to solve them.

Chapter 7 introduces MAB-DCOPs and describes the UCB-GDL class of algorithms to solve them.

Chapter 8 concludes and discusses future work.

Chapter 2

Related Work

As stated in Chapter 1, the research presented in this thesis is both practical and theoretical. The aim of this chapter is then to analyse and discuss practical and theoretical research related to the work presented here. Thus, we will discuss both the research that has previously attempted to meet the requirements of our setting, and also the research that we will use as the foundation over which to build our contributions.

In more detail, Sections 2.1 and 2.2, respectively, discuss unmanned aerial vehicles (UAVs) and disaster response problems related to situational awareness. As discussed in the previous chapter, the aim of this thesis is to propose *general* algorithms that can be applied to coordination problems pertaining to multiple application domains. However, to study the performance of the previous algorithms we chose the domain of situational awareness for disaster response in consideration of its complexity and the numerous challenges that make it extremely compelling for coordinating autonomous vehicles. Similarly, we chose to study coordination when applied to UAVs, in recognition of their information gathering capabilities. In light of this, Section 2.1 presents different categories of UAVs and describes their main design features such as their flight capacity, the altitude that they can reach and their degree of autonomy. Section 2.2 then discusses the type of research problems related to disaster response. In fact, there exists an extensive literature on coordinating UAVs to achieve situational awareness. We identify here four different problem categories and describe them in detail. At the end of both sections, we highlight the way in which we use this background for the purposes of our work.

The remainder of the chapter then focuses on coordination. In particular, we present an extensive description of coordination algorithms. We distinguish here between approaches that have been used in practice and those that are still being studied from a theoretical perspective. Within both categories, we present the approaches that are most relevant to our settings. In more detail, Section 2.3 presents coordination approaches

which have been used in practice. That is to say, they have been used to solve some of the problems described in Section 2.2. We describe how these coordination algorithms are typically embedded within a generic decentralised decision making architecture that is used to coordinate the actions of the UAVs. This system also comprises a belief formation and a navigation module which are used to define, respectively, the knowledge over which to make a decision and the actions corresponding to each decision made.

Section 2.4 defines the approaches studied from a theoretical perspective. This means that these approaches have not yet been deployed, since some of their theoretical properties, such as quality of the solution recovered, computation and communication required, are still being investigated. To represent the problems in Section 2.2, we will adopt the framework of distributed constraint optimisation problems (DCOPs). The reason for this and the actual framework are detailed in Section 2.4.1. In essence, we selected DCOPs because they constitute a good compromise between the expressiveness (i.e. the capability to represent the problems' constraints) required by the problems we wish to solve and the computational complexity necessary to solve them. Next, canonical DCOP algorithms are described. These are the most common type of approaches in the literature and, in few words, consist in optimising a real valued function in a decentralised fashion. Among these, we focus specifically on algorithms that exploit the generalised distributive law (GDL). This framework enables algorithms to exploit the factorisability of a broad class of optimisation problems (which includes DCOPs) to solve them in an effective and efficient manner. This factorisability is typically encoded within the algebraic structure of the global function to optimise. In particularly, as we will see for DCOPs (Chapter 2.4.1), it can be defined as a *commutative* semi-ring. By exploiting this structure then, GDL algorithms have been successfully applied to a variety of different domains such as information theory (MacKay, 2003, 1999), artificial intelligence (Frew et al., 2008; Frey et al., 1997) and statistical physics (Mezard et al., 2002). We focus here on describing how the GDL can be used to derive algorithms capable of solving DCOPs. Next, we move our focus onto literature that addresses problems involving complex agents interactions, as discussed in Chapter 1. In particular Section 2.4.3 describes algorithms to solve optimisation problems involving complex functions involving multiple objectives, uncertainty and online learning. As will be described in more detail in the following chapters, these problems present all the same features, that the global function to optimise is defined over a partial order. In conclusion of the chapter, Section 2.5 summarises the chapter and states the background that will be used throughout the rest of this thesis.

2.1 Unmanned Aerial Vehicles

Recent years have seen UAVs becoming one of the most studied and deployed type of robotic agents. Their effectiveness at achieving accurate situational awareness in open environments makes them very useful for a variety of real world application domains, both civil and military. Examples of military operations include reconnaissance, surveillance, target acquisition and meteorology missions (Sarris, 2001). Examples of civil operations include search and rescue, communications relay and disaster and emergency management (Sarris, 2001; Valavanis and Valavanis, 2007). As a consequence, it has been estimated that the use of this type of agent will increase in the next years (UAS Roadmap, 2010).

Due to the variety of operations that UAVs can be used for, various types have been developed. Generally, UAVs can be either fixed or rotary wing (see Figures 2.1(b) and 2.1(a) showing an example of each). Fixed wings UAVs are typically used for surveillance and target acquisition missions. However, they are not suitable for collecting live imagery since they are not capable of hovering— stationing above a specific location. In contrast, rotary wings UAVs can do so and for this reason they are typically deployed for missions such as situational awareness or reconnaissance, and will be considered for our setting.

UAVs can be classified based on the altitude that they can reach, their endurance (i.e. their battery capacity) and their communication range (Sarris, 2001). To provide some examples, Figure 2.1(a) depicts one of the largest UAV developed to date; the RQ4 Global Hawk intended for intelligence collection and surveillance missions, and thus provided with high quality sensing devices. This platform is considered a high altitude (over 9000 m), long endurance and indefinite range vehicle, designed to provide wide area coverage of up to 40,000 m^2 per day. Global Hawks are currently being used by the United States Navy for maritime surveillance operations (UAS Roadmap, 2010). Figure 2.2 shows a picture of the MQ8 Fire Scout, a medium-altitude (up to 9000 m), long-endurance (over 200 km) UAV, widely used for reconnaissance and situational awareness missions in Iraq, Bosnia and Afghanistan by the United States Air Force.

Additionally, UAVs can be classified depending on the way in which they are controlled. Two possibilities exists, "drones" (Figures 2.2, 2.1(a) and 2.1(b)) are controlled by a human pilot which operates from the ground. Thus far, these are the most reliable vehicles, and the ones that are deployed the most in real missions. In contrast, fully autonomous vehicles are controlled by a decision making system completely independent from any human operator. Figure 2.3 shows an example of such vehicle. Fully autonomous vehicles are still being tested, and very few have been deployed on real missions. However, they are advocated as the key vehicle to deploy for situational awareness operations in the future (Bethke et al., 2008).

(a) The RQ4-Global Hawk, a remotely controlled, "drone" used by the US-army.

(b) The A160 "Hummingbird" a rotary wing "drone" developed by Boeing.

FIGURE 2.1: Two different types of UAVs.

The above mentioned UAVs are typically deployed for military applications. For this reason, they are extremely sophisticated and expensive vehicles. In contrast, research and civil industry do not typically require such a level of complexity and have used cheaper and more customisable vehicles. A number of these vehicles are depicted in Figure 2.4. These vehicles are known as micro-UAVs, albeit in recent years this definition has been attributed to even smaller vehicles measuring few centimeters, and are used for a wide range of applications from surveillance to scientific research. Thus, we will consider this type of vehicle for our practical contributions. In more detail, we will consider two commercial off-the-shelf Mikrokopter hexacopter rotary wing UAVs (see Figure 2.5). These are a type of small UAVs, currently used by many law enforcement

FIGURE 2.2: The MQ-8 Firescout a remotely controlled UAV used by the US-army.

FIGURE 2.3: The BAE-Herti, a fixed wing UAV developed at BAE Systems.

agencies for situational awareness operations (How et al., 2009; Bethke et al., 2008). In addition, these vehicles are also used for research purposes. Typically, simple vehicles are used which can reach an altitude up to 40m and whose battery can last up to 15 minutes. Each vehicle localises its position using a global positioning system (GPS) receiver and monitors its status through a set of sensors including an inertial measurement unit, a magnetometer and a barometric altimeter. For security reasons, we implemented each vehicle so that it could be controlled either by a human operator (for takeoff and landing) or an autonomous controller.

FIGURE 2.4: An illustration of different types of micro-UAVs

FIGURE 2.5: The "Hexacopter" UAVs used in the flight tests presented in Chapter 3.

2.2 Coordination Problems for Disaster Response Scenarios

The main application domain of this thesis is that of disaster response. In our case, we wish to support first responders at the scene of a disaster to provide them with accurate situational awareness to prioritise intervention.

To achieve this awareness, UAVs are typically deployed to collect and process information about the area. This information typically defines some features of the environment. In the context of disaster response these features can be any piece of information relevant to the first responders at the scene of the disaster. Examples include the position of casualties, the radiation level emanating from a container or the temperature of a burning building. Each deployed agent is then provided with a sensor to collect information about such features. Examples include bearing-range sensors which measure the distance between the UAV and the feature, cameras used to collect live aerial imagery

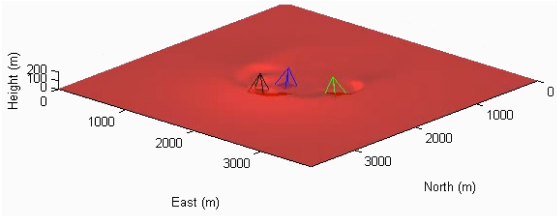


FIGURE 2.6: A snapshot of a PDA's interface depicting three UAVs collecting information

from the environment, and thermal or radiation sensors able to capture the temperature or the level of radiation emanating from a specific environment (Xu and Zhang, 1996). To visualise this information, first responders could use a personal digital assistant (a PDA) which would describe the amount of information collected by the UAVs. Figure 2.6 shows a possible way to represent such information. In the figure, three UAVs (blue, green and black) operate within a squared area and make observations to reduce the uncertainty (i.e. maximising the amount of collected information) about the feature described previously.

Given the specific type of feature, four different classes of problems can be identified. Each of these classes constitute one challenging information gathering problem studied for disaster response. Moreover, each class represents a research problem in which the deployed agents would greatly benefit from adopting a coordinated behaviour. Hence, in the remainder of this section, we present each of these classes to give the reader an idea of the type of settings in which robotic agents are typically deployed and for which coordination algorithms need to be derived.

However, the focus of this thesis is on studying coordination approaches to the decision making capabilities of the former agents. Hence, within our practical contributions we will consider a situational awareness problem, in which the UAVs do not have to process the information that they have collected but simply report it back to the first responders (see Chapter 3 for more details). In so doing, we will abstract out the challenges related to build a consistent estimate of the feature that is being explored and focus only on the coordination algorithm. Similarly, for our theoretical contributions we will not study the former problems specifically, but will consider them as the key settings where to apply our algorithms for future work. In fact, most of the problems that we are going to present constitute motivating scenarios for our algorithms. Examples will be presented within each specific section.

2.2.1 Target Search Problems

These problems deal with searching a possibly unknown and hostile environment for one or more targets whose position is unknown (Waharte and Trigoni, 2010). In the

context of disaster response, a typical motivating scenario might consist of a team of UAVs deployed after the sinking of a ship to search for one or more drifting life-rafts whose position is unknown (Bourgault et al., 2003, 2004). Within this setting, the aim of the team of UAVs is to detect the target(s) in the smallest amount of time.

Bourgault et al. (2003) represented the problem of searching for a single target as an information collection and fusion problem. Specifically, a Bayesian estimation approach was used, in which a probability grid was used to represent the possibility of the target being located within any discrete area of the search space. Subsequent research has investigated a range of variants of this problem changing the number of UAVs and targets, the scenario settings (e.g. an urban or maritime search task), or the way the probability over the location of the target is represented. For example, Cole (2009) addressed the search for groups of targets when their total number is unknown. Geyer (2008) considered more complex search areas such as an urban environment, whilst Vidal et al. (2001), Yang et al. (2005) and Waharte et al. (2010) addressed search in hostile environments containing threats and obstacles respectively.

In this work, various degrees of abstraction have been used to formalise these different settings. Some frameworks abstract the real world setting by representing the searching area as a graph (Hoffmann et al., 2006) or as a grid (Hespanha et al., 1999; Vidal et al., 2001; Antoniades et al., 2003). In the latter setting, the agents, representing the UAVs, move by simply "jumping" from one node to another. The agent's perception then consists of observing the status of some of the nodes in its proximity to see whether they are occupied by a target or not (Hespanha et al., 1999; Vidal et al., 2001; Antoniades et al., 2003). More realistic frameworks have also been adopted, in which the area of interest is represented as a continuous space where each agent moves using a continuous motion model based on speed and heading (Bourgault et al., 2003, 2004; Cole, 2009).

For the sake of our work, we will use target search problems as a motivating domain for studying coordination under uncertainty. Indeed, in target search uncertainty is endemic, since sensors are noisy, the environment is unknown and the targets' position is most likely unknown with precision. This means that the outcome of the agents decisions is defined by a pdf which needs to be taken into account in their decision making process. Moreover, as we discussed in Chapter 1, addressing such uncertainty requires taking into account the risk associated with the agents' decisions (i.e. their exposure to an undesirable outcome). In so doing, the agents are now making complex decisions for which new coordination algorithms need to be derived, as we will discuss in Chapter 6.

2.2.2 Target Localisation and Tracking Problems

These problems involve tracking a target within an unknown and possibly hostile environment in order to get a precise measure of its position. A typical motivating scenario follows from the one described earlier where, after the sinking of a ship, a team of UAVs is deployed to continuously track the position of the drifting life-rafts to transmit their location to the first responders (Furukawa et al., 2006). Within this setting, the aim of the team of UAVs is to localise the target's position with the greatest precision in the smallest amount of time (Symington et al., 2010). Similarly to target-search, the principal research challenges for target tracking involves varying the number of UAVs (Stolle and Rysdyk, 2003) and the number of targets (Frew et al., 2008), which can be either static or moving (Symington et al., 2010; Cole, 2009). Further issues have also been considered to make the setting more realistic. Examples include modelling the wind preventing the vehicles from maintaining the position of the target or considering the presence of obstacles or threats (Ousingsawat and Campbell, 2004; Zengin and Dogan, 2007).

In addition, current research is exploring settings in which the UAVs are tasked to track large groups of targets. In disaster scenarios, these groups can typically represent humans (civilians), animals or vehicles, that are either moving towards a refuge, evacuating or running from a building on fire. Initial work has focused on tracking groups of ground targets, but without explicitly considering the dynamics of group motions and behaviour (Sommerlade and Reid, 2008). However, taking into account the motion and the behaviour of the former groups has been shown to be the key challenge to address within these settings. Hence, clustering algorithms have been proposed to group targets with similar motion (Ma and Zhang, 2004).

Similarly to target search problems (Section 2.2.1), we will use target tracking problems as a motivating domain for studying coordination under uncertainty. Indeed, in target tracking uncertainty is endemic, since sensors are noisy, the environment is unknown and the targets' position is most likely unknown with precision. Hence, again the uncertainty and the risk associated with their decisions need to be taken into account, as we will discuss in Chapter 6.

2.2.3 Search and Tracking Problems

These problems encompass both the problems presented in Sections 2.2.1 and 2.2.2. A motivating scenario is a hybrid between those presented above; a team of UAVs is deployed after the sinking of a ship to search for the drifting life-rafts, while tracking the positions of those that have already been found. The problem has been formalised by Furukawa et al. (2006) for the single target case. In this case, the problem consists

of a sequence of search and track phases. The UAVs search for the target, track it, once it is found, and the whole process is repeated whenever the target is lost, until sufficient information has been gathered. It was subsequently extended to multiple targets by Furukawa et al. (2007). This scenario can be seen as a target assignment problem where the team of UAVs cooperate to achieve two different objectives:

Searching: Part of the team searches for new targets, or for those targets that have been lost.

Tracking: Part of the team tracks those targets that have been found in order to maintain a precise estimate of their position.

Thus far, approaches for search and tracking problems have been mostly extensions to the ones used for the standard search problem. In particular, Furukawa et al. (2006) characterise the problem as one of information gathering and present a unified approach for solving it. The same representation of the belief over the target position is used for both searching and tracking. Specifically, a probability grid, similar to the one described in Section 2.2.1, is used and different types of observation updates are defined for the tracking task (i.e. when the target has been detected) and for the search task (i.e. when the target has yet to be detected). A similar approach was used by Tisdale et al. (2008) who further included the probability of false detection (i.e. sensors are not perfect). In this same work, a comparison is made between the two principal approaches used to represent the probability density function (pdf) representing the uncertainty about the target's location. In more detail, approaches that consist of sampling the pdf (i.e. particle filters) are compared against probability grids in which the pdf is discretised. The latter were shown to be as effective as sampling based approaches for the search task. However, the higher degrees of accuracy required for estimating the target's positions require the use of particle filters. A different approach, the element based method, is proposed by Furukawa et al. (2007) in which the former pdf is modelled as a graph. By so doing, Furukawa et al. (2007) shows that his method is more efficient in terms of computation required to represent the pdf, than grid-based methods.

A task assignment formalisation of the search and track problem was introduced by Drew and Elston (2008). In their approach, each UAV maintains an area coverage map representing information about the targets and a central controller determines which targets need to be re-observed in order to decrease the uncertainty about their position to some desired bound. Once it has been decided which UAV should be tasked to track the targets, the remaining agents are devoted to searching. A similar approach is proposed by Jin et al. (2004), where the search and track problem is translated into a military domain for searching, confirming and attacking some specific targets within an area. Here a completely centralised architecture is proposed where a central controller

acquires all the information gathered by the different UAVs and assigns the different tasks to each of them.

For the sake of our work, we will use search and track problem as a motivating domain for studying coordination over multiple objectives. As we have seen, these problems are defined by two objectives (i.e. search and track), which conflict upon the allocation of the agents (see Chapter 1). This means that the agents are now making complex decisions over the former objective, for which new coordination algorithms need to be derived, as we will discuss in Chapter 5.

2.2.4 Airborne Simultaneous Localisation and Mapping Problems

Canonical simultaneous localisation and mapping (SLAM) problems refer to the case in which teams of autonomous vehicles are deployed within an unknown environment to build a map, whilst simultaneously keeping track of their own position (Thrun et al., 2005). One of the many motivating scenarios for this problem follows from the one depicted in Chapter 1 where unmanned vehicles are deployed to map the interior of buildings at a disaster scene, in order to provide the first responders with a precise map of each building, with the position of all the casualties within each appropriately marked. The SLAM aspect is typically addressed by defining an initial level of uncertainty over the map to be built, and subsequently attempting to minimise this uncertainty by choosing appropriate new observations until a predefined threshold is reached. In practice, the problem is often solved using data fusion techniques such as Kalman and particle filters or Gaussian processes (Thrun et al., 2005).

Sukkarieh and Durrant-Whyte (2001) have brought the SLAM problem, which was previously studied considering unmanned ground vehicles (UGVs) only, to settings involving UAVs tasked with surveillance (Kingston et al., 2008) and planetary exploration tasks (Braun et al., 2004) in which GPS position is unavailable. The key difference in such settings, is that UAVs perceive the environment differently than UGVs, since not only do they have different sensors, but they also take aerial instead of ground observations. This means that the methods to represent the map of the environment need to be changed. For this reason, Sukkarieh and Durrant-Whyte (2001) focused on developing techniques to improve the quality of the resulting map by adapting the previously mentioned data fusion techniques to UAVs with different type of sensors and motion dynamics (Bryson and Sukkarieh, 2007; Caballero et al., 2006). The same problem has been extended to settings involving multiple UAVs by Kim and Sukkarieh (2003), which also describe various decentralised data fusion techniques that allow the UAVs to map the environment in a consistent fashion. The benefit of adopting a cooperative approach have been studied by Bryson and Sukkarieh (2007) and it was shown how the UAVs could improve the accuracy of their maps by adopting coordinated behaviour. However, to date the

problem of studying SLAM for UAVs is still relatively new. Consequently it constitutes an open problem requiring further studying.

For the sake of our work, we will use the SLAM problem as a motivating domain for studying coordination and online learning. Indeed, in these problems, agents not only have imprecise sensing capabilities, but are also deployed without any prior information about the environment where they are located. In this setting, learning the outcome of the agents' decisions is a key requirement for improving their decision making process. Indeed, the agents are now making complex decisions trading off their estimate and the uncertainty associated with it. Hence, new coordination algorithms need to be derived, as we will discuss in Chapter 7.

In addition, to deploy these algorithms they need to be properly embedded within the control system of each UAV. This system is typically composed not only by a decision module, where the coordination algorithms is implemented, but also by a sensing, a communication and a belief formation module. Hence, to deploy such algorithms, research in applied robotics has initially studied the way in which they can be deployed within such systems. In fact, a generic coordination architecture has been derived, which we discuss next, along with the different approaches that have been deployed.

2.3 The Practice of Coordination Algorithms

Coordination approaches applied in practice have been studied mainly within the robotics community. Hence, a vast literature exists on these algorithms (Dias et al., 2006; Bethke et al., 2008). This thesis focuses on decentralised approaches, since they do not rely on a central point failure and are, as a consequence, robust against the failure of one or more agents (Chapter 1).

Now, current research typically considers these algorithms to be part of a generic decentralised architecture that is used to coordinate the decision making process of the UAVs whilst deployed to solve the problems presented in Section 2.2. Figures 2.7, 2.8 and 2.9 illustrate this architecture. In more detail, the figures show how the architecture is modeled as a protocol defined by the following modules:

Sensing : This module is responsible for elaborating the measurements of the environment made by each UAV's sensors and transmitting them to the belief formation module in the form of an observation that can be processed (Xu and Zhang, 1996).

Belief Formation : This module is responsible for processing an agent's observations and fusing them together into consistent beliefs about the features of the environment that are being observed (Grocholsky, 2002).

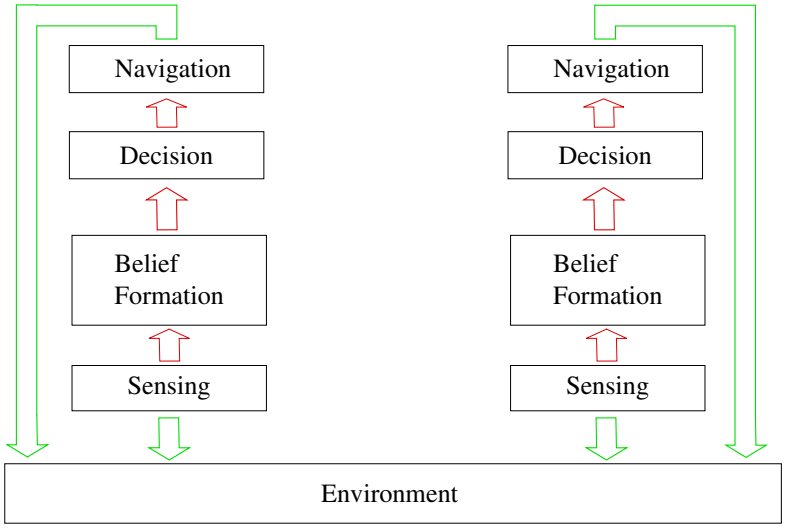


FIGURE 2.7: The architecture of a non-coordinated approach (Cole, 2009)

Decision : This module is responsible for making a decision given the belief information that the agent was able to build (Cole, 2009).

Navigation : This module takes as inputs the outputs of the decision module. In essence, it transforms the decisions made by the UAVs into actual manoeuvres.

Now, this architecture is implemented within each UAV and determines its behaviour throughout an operation. In our work, we will use this architecture for deploying our coordination approach on real UAVs, as we will see in Chapter 3. Indeed, this architecture constitutes a fundamental framework over which to design, implement and deploy coordination algorithms for UAVs. In addition, literature has shown that a coordinated behaviour can be imposed by having the specific modules of each UAV communicate between each other (Cole, 2009; Grocholsky, 2002). By using this fact, it is possible to classify decentralised coordination approaches for UAVs into three broad categories, based on the type of information shared by the different agents. In our work then, we will use one category to build the control module of our UAVs, use another one as a benchmark, and will consider one for future work. In the remainder of this section, we describe each of these categories and the way we are going to use them.

2.3.1 Non-Coordinated Approaches

This category encompasses all approaches in which the agents behave independently, and do not interact in any way with the other members of the team. Figure 2.7 illustrates the canonical architecture for a non-coordinated approach, where each agent makes its own decisions independently, without sharing any information with the other members of the team.

Due to the computational requirements of each of the different modules, greedy techniques are commonly considered the best method to use to define non-coordinated approaches. By using such approach, each agent makes a decision given only the local information that it has collected thus far (Vidal et al., 2001; Antoniades et al., 2003). The key challenge of this approach is to define techniques that allow the agents to maintain and refine a highly accurate estimate of the features that are being measured (Section 2.2). Research in both robotics and artificial intelligence has typically used probabilistic reasoning techniques to represent the information about such feature. For example, in the context of UAV operations, Bayesian filters have been advocated (Thrun et al., 2005). These filters define a pdf over the state of the features and then progressively update them as new observations are made by the UAV. Bayesian filters are computationally efficient due to their iterative nature and yield accurate measures for all the problems described in Section 2.2. A wide variety of such filters exist. Examples include the Kalman, the particle and the information filter, which can be used depending on the specific type of pdf that is being considered (Thrun et al., 2005).

Bayesian filters can also easily provide estimates of the future state of any feature. To do so, a prediction step is added to the algorithm in which the learned model of any feature is run forward in time. For instance, in the context of target tracking or searching, the prediction step can be used to estimate the next area or location where the target is going to be. This prediction step is extremely useful for decision making since it allows the effects or the impact of an agent's decisions to be estimated. For this reason different time horizons have been considered in the literature, depending on how much computation would be required by longer time intervals[1]. Typically, myopic or fixed time horizon predictions are considered in real applications (Hoffmann et al., 2006).

The belief built through these filters is then used to make a decision by defining a utility function, based on the predicted pdf, and defined over all the possible decisions that the agent can make. In the context of UAVs, decisions can be areas to explore, trajectories to take or tasks to complete. The utility functions are typically problem-specific and depend on the features that are being measured. For instance, in target search the utility function may be defined as the probability of detecting the target in a specific sub-region of the search space (Bourgault et al., 2003, 2004). Conversely, problems such as target tracking use canonical information metrics, such as the Shannon entropy or the mutual information gain, to measure the reduction in uncertainty that results from the observation of a specific target's position (Cole, 2009).

Now, the previous discussion describes independent behaviours for each agent. For the sake of our work, we will use a non coordinated approach as a benchmark for our coordination mechanism in Chapter 3. Indeed, non coordinated approaches constitute a

[1]This type of method is also referred as Model Predictive Control in related literature (Camacho and Bordons, 2003; Garcia et al., 1989)

very good lower bound on the performance of coordination algorithms. However, when a team of agents is deployed, each agent should consider the behaviour of the other members of the team to increase the overall performance. In fact, non-coordinated approaches are likely to result in very poor performance when multiple agents are deployed and more sophisticated approaches are necessary.

2.3.2 Implicitly Coordinated Approaches

This category encompasses all approaches in which the agents share their observations, but make decisions independently. By sharing observations, each agent is able to build the same common model about the state of the features that are being measured, thus ensuring more accurate information over which it will take its decision (Grocholsky, 2002). Hence, the agents are able to make decisions based on information that they have not themselves collected from the environment. This is typically done using decentralised data fusion techniques; a well known and well studied set of techniques to efficiently estimate the state of a feature of interest by using Bayesian estimation (Bourgault et al., 2003, 2004; Hoffmann et al., 2006; Cole, 2009). In more detail, each UAV collects information locally and fuses it together with the information acquired by the other agents. The aim is to build a consistent global view of the feature of interest.

In essence, each UAV builds a local estimate of the features that are being measured, using Bayesian estimation techniques as described in Section 2.3.1. However, in this case, each UAV also incorporates the observations of the other agents while updating the pdf of the features of interest. Decentralised data fusion techniques then work by continuously refining this estimate by incorporating all the UAVs' new updates (Thrun et al., 2005). Hence, as shown in Figure 2.8, the agents share their observations, adding a message passing phase in the belief formation module of their coordination architecture (Grocholsky, 2002).

The decision module of an implicit coordination approach typically uses the same decision making techniques as the non-coordinated approaches, since the main difference between the two levels lies essentially in the belief formation module, where the agents share their observations.

However, by having global information about the feature that is being measured, the agents are able to determine the decisions that will allow them to minimise the uncertainty about the feature that they need to measure the most. As an example, in the target-search problem, the agents will aim for the unexplored sub-regions of the searching space (Bourgault et al., 2003). In our work, we will not consider an implicit coordination mechanism to benchmark our approach. The reason is that in our work, as we discussed in Section 2.2, the first responders will be responsible for processing the

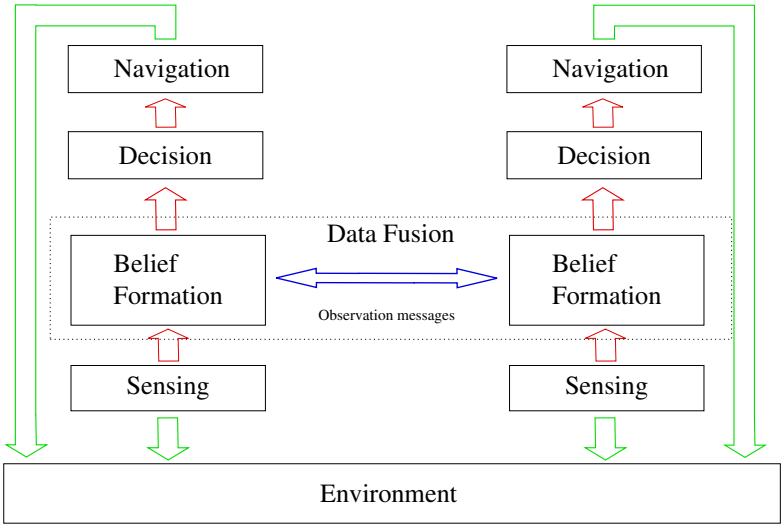


FIGURE 2.8: The architecture of an implicit coordinated approach (Cole, 2009)

information captured by the UAVs. However, in so doing, as we will discuss in more detail in Chapter 3, each first responder obtains only the information that he requested. Consequently, each first responder only collects a portion of the information of a disaster scene. As we have explained in this section, this limits the decision making capacities of the UAVs. Hence, we envisage in our future work (see Chapter 8) to extend our approach to provide each first responder with a global information of the scene.

The main disadvantage of implicit coordination is that by deciding over the same knowledge, the agents are likely to make the same, redundant, decisions (e.g. explore the same area). This redundant behaviour is clearly not advisable in problems where timeliness is essential, such as target search or target tracking (Hoffmann et al., 2006). Hence, to perform effectively, the agents need to take into account the decisions of the other members of the team explicitly. To this end, we discuss explicit coordination techniques next. As we will see, the use of explicit coordination greatly improves the quality of the agents' decision making. However, this often comes with a large computational overhead, thus implicit coordination has been shown to be useful in many robotics applications such as target search and target tracking (Hollinger et al., 2009).

2.3.3 Explicitly Coordinated Approaches

As discussed above, this category encompasses approaches where the agents share both their decisions and their observations. Figure 2.9 shows an example of such an approach, where both levels of information sharing are illustrated. The process of sharing observations is carried out in the same fashion as for the implicit coordination case. However, in the explicit coordination case, the agents make their decisions by taking into account the actions of the other members of the team (Bethke et al., 2008). The main drawback

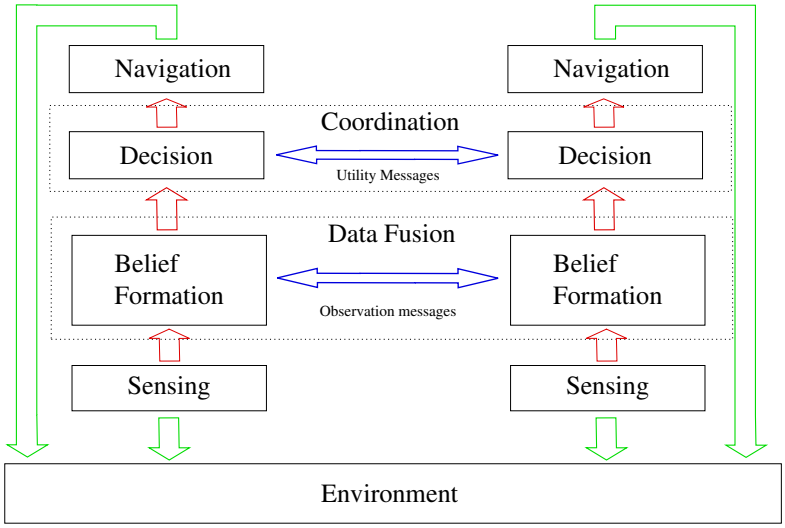


FIGURE 2.9: The architecture of an explicit coordinated approach (Cole, 2009)

of explicit coordination techniques is that finding a joint optimal decision for a team of agents is computationally expensive (exponential in the number of agents) (Hoffmann et al., 2006), and thus the problem rapidly becomes infeasible as the number of members of the team grows larger.

To address this shortcoming, the robotics community has proposed a variety of decentralised approximate algorithms which compute approximate solutions by exploiting the locality of the agents interactions (Bethke et al., 2008). The key idea is for an agent to share its decisions with those other agents whose behaviour is more likely to affect the former decisions. The latter agents are typically in the former's immediate proximity (i.e. in its neighborhood). These algorithms are the most suitable for real world domains since they have been shown to be computationally efficient given the agents limited computational and communication resources. Typically, they work as a negotiation protocol in which the agents share their decisions for a pre-specified amount of time, and then each agent selects its decision as the best response to all the possible decisions of its neighbourhood (Alighanbari and How, 2005, 2006). The main drawback of these approaches, however, is the quality of the solutions they find, which can be very poor and can often require multiple messages to be shared (Grocholsky, 2002).

Another category of negotiation approaches is the one of market based allocation algorithms (Dias et al., 2006; Ham and Agha, 2008). In essence, by using these techniques, each agent participates in a sequence of auctions, in which it bids for the specific decision that it wants to make considering its own various constraints, such as its remaining battery capacity and the benefits that it would get from making such decision. These techniques meet many of the requirements defined in Chapter 1, since they are very efficient in terms of both computation and communication. However, most of these algorithms are either centralised or partially centralised, since one of the agents always

needs to act as the auctioneer (i.e. the agent retrieving the bids and determining the winner who is assigned the task). Hence, there is often a possible central point of failure which makes these algorithms not suitable for our settings. Furthermore, fully decentralised auction mechanisms, in which each agent acts as a bidder and as an auctioneer at the same time, suffer from the same drawback of the negotiation algorithms described earlier (How et al., 2009).

In our work, we will use an explicit coordination mechanism to deploy the max-sum algorithm, which we are going to present in Section 2.4.4. The reason comes from the advantages for this type of architecture to deploy decentralised decision making mechanisms, which we just discuss. However, as we will see in Chapter 3, depending on the specific problem, there exists multiple ways of deploying max-sum. For this reason we will also present a methodology guiding a user to the successful deployment of the algorithm for coordinating unmanned vehicles. The reason why we chose to use the max-sum algorithm then will be presented in the next section. In essence, research in robotics has focused more on implementing new techniques for fusing information in a consistent fashion. For this reason, research on coordination algorithms has been limited to deploy well known negotiation protocols such as those discussed in this section. In contrast, the artificial intelligence and the multi-agent systems communities, have studied coordination algorithms to address more complex interaction scenarios, despite doing it only in simulation. These approaches are the focus of the next section.

2.4 The Theory of Coordination Algorithms

The theoretical study of coordination approaches has principally been undertaken by the artificial intelligence and the multi-agent systems communities. Again, we focus our attention on decentralised approaches, since they do not rely on a central point failure and are, as a consequence, robust against the failure of one or more agents (Chapter 1).

However, before describing these algorithms, we need to select a framework to represent the problems that we wish to solve. A number of such frameworks exist, which have been applied to domains such as disaster response (Dias et al., 2006) or energy (Vytelingum et al., 2010). However, as we discussed in the previous section, the algorithms used to solve these problems typically yield a poor performance in terms of improving the agents' decision making capabilities. Hence, we focus here on more theoretical frameworks. Sequential decision making frameworks such as Markov decision processes (MDPs) or partially observable Markov decision processes (PO-MDPs) are often advocated (Guestrin et al., 2003; Littman, 2009). These frameworks represent coordination problems as a sequential Markov decision process in which the agents have to make a new decision at each time step. In essence, the agents coordinate to determine

a sequence of decisions (i.e. a plan) that each agent has to make, considering its present and future states. However, these frameworks have also been shown to be extremely expensive in terms of computation (Bernstein et al., 2000). For this reason they do not allow solution techniques to scale to a reasonable number of agents. Scalable techniques, based on very strong approximation techniques such as monte Carlo sampling, have been proposed to address this issue (see for instance (Doshi and Gmytrasiewicz, 2009), (Roy, 2003) and (Amato, 2010)). However, to achieve feasibility, these techniques sacrifice the quality of the solutions recovered. Thus, they violate the requirement of performance guarantees (see Chapter 1) and thus, they are not suitable for our setting.

In contrast, distributed constraint optimisation problems (DCOPs) represent coordination problems as "one shot" optimisations, and for this reason they constitute a good compromise between quality and feasibility (Modi et al., 2005; Rogers et al., 2011). Specifically, the DCOP framework has been shown to be very promising for representing many coordination problems related to the real world, such as disaster management, meeting scheduling and sensor networks (Chapman et al., 2011; Sultanik et al., 2007; Rogers et al., 2011). For this reason, in recent years both the robotics and the artificial intelligence communities have increased their interest in this framework. To this end, we describe the DCOP framework in Section 2.4.1. In Section 2.4.2, we then describe canonical DCOP algorithms. More precisely, we will discuss optimal, approximate and bounded approximate algorithms. These algorithms solve problems in which the value of the global constraint function is real valued. Consequently, the solutions of a DCOP can be *totally ordered* (i.e. there exists a binary dominance relation $<$, which can be used to order all the different values of the global function) and a single global optimum (maximum and minimum) exists. However, as we explained in Chapter 1, a single real valued function is not sufficient to represent complex agents' interactions which is one of the key aims of our work. For this reason, in Section 2.4.3, we discuss algorithms capable of addressing complex optimisation problems involving multiple objectives, online learning and uncertainty. A common feature of these algorithms is that they all solve optimisation problems defined by *partially ordered* solutions. The partial order comes from the fact that to rank these solutions it becomes necessary to trade off between different, possibly conflicting, parameters such as the various objectives or the mean and the variance defining the uncertainty. As a consequence, a single optimum might not exist anymore, rather we might have a set of equivalent *non dominated* solutions. We are interested in these algorithms since they represent possible solutions to the type of complex coordination problems that we wish to model in this work (Chapter 1). In some cases, they also constitute the foundation over which we will build our algorithms in the following chapters. In Section 2.4.4, we then describe the GDL framework and the way it can be used to derive new algorithms. The key advantage of GDL algorithms is that they exploit the algebraic structure of the global function of an optimisation problem (i.e. it is characterised by a commutative semi-ring) to solve the problem in an

efficient manner (see Section 2.4.1). Existing algorithms then exploit this property to solve optimisation problems (including DCOPs) efficiently. In light of this, we are going to use it to build our new algorithms in Chapters 4, 5, 6 and 7.

2.4.1 Distributed Constraint Optimisation Problems

A DCOP is a generic framework to represent a multi agent coordination problem. Formally, a DCOP is defined as follows:

Definition 2.1. A DCOP is a tuple $\langle A, X, D, F, C \rangle$ where:

- $A = \{1, \ldots, |A|\}$ is a set of agents.and the algorithms that are based on it
- $X = \{x_1, x_2, \ldots, x_n\}$ is a set of variables.
- $D = \{D_1, \ldots, D_n\}$ is a set of discrete domains, where D_i is the domain of variable x_i.
- $F : X \rightarrow A$ is a function that assigns variables to agents. Each agent controls (the value of) the variables that are assigned to it.
- $C = \{C_1, \ldots, C_m\}$ is a set of local constraint functions. Each C_j $(1 \leq j \leq m)$ is defined over local scope $X_j \subseteq X$, and assigns a real value to each assignment of X_j. The domain of set X_j is D_{X_j}, which is the Cartesian product of the domains of variables in X_j.

The solution of a DCOP is an assignment X^* to variables X that maximises the global constraint function:

$$X^* = \arg\max_X \sum_{j=1}^{m} C_j(X_j) \tag{2.1}$$

A DCOP is often represented as a constraint graph between the agents. In such graph, the nodes represent the variables in X and the edges represent the constraints $C_j \in C$ between such variables. Typically, to represent $n-$ary constraints the use of factor graphs has been advocated (Kschischang et al., 2001). A factor graph is an undirected bipartite graph in which vertices represent variables $x_i \in X$ and constraints $C_j \in C$, and edges encode the relation "is a parameter of" between a variable and a constraint (Kschischang et al., 2001). An example of a factor graph is shown in Figure 2.10. The factor graph in the figure describes a DCOP containing two agents A_1 and A_2. Each agent is assigned one variable and one constraint. A_1 is assigned variable x_1 and constraint $C_1(x_1, x_2)$ and A_2 is assigned variable x_2 and constraint $C_2(x_1, x_2)$.

By using such graphical representation DCOPs can encode coordination problems such as those defined in Section 2.2. An intuition to understand this is that a factor graph is, essentially, a graphical interpretation of the global constraint function defined by Equation 2.1. Thus, since in a coordination problem the agents contributions and the problem's constraints are typically encoded within such function, the use of a factor graph allows to decentralise all such contributions and constraints between the different nodes of the graph. In so doing, it becomes possible to precisely define the contribution of each agent to the solution of the problem.

As discussed in Section 2.2, this might be quite useful to deploy agents to achieve situational awareness. For instance, consider a multi target tracking problem (Section 2.2.2) consisting of two UAVs and two targets. This problem can be easily encoded in the factor graph presented in Figure 2.10: the UAVs are encoded in the variables x_1 and x_2, whereas the utilities for tracking the two targets, namely T_1 and T_2, are encoded in the constraints C_1 and C_2. Each variable represents the decisions of one agent and is defined over the different targets that the agent can track (i.e. T_1 or T_2). The objective of the agents is to jointly decide which target each of them should track and is encoded in Equation 2.1.

As a second example, consider a target search problem (see Section 2.2.1) involving two UAVs searching for a lost target. This problem can also be encoded in the factor graph of Figure 2.10. More specifically, the two constraint functions C_1 and C_2 represent the agents utilities (i.e. their contribution to the search objective) and the variables x_1 and x_2 represent the agents manoeuvres (e.g. the bank angles of the UAVs). The agents' objective is then to move into the search area and make observations to reduce the probability of detecting the target represented by Equation 2.1.

Unfortunately, not all aspects of a coordination problem can be mapped within a DCOP or into a factor graph. In general, all aspects that cannot be encoded within the constraint functions cannot be encoded in a DCOP. Examples include time deadlines or budget constraints, which are present in numerous real world problems. Most likely these constraints are present in problems related to situational awareness. For example, consider a team of UAVs tasked to search a certain area until a limited time (i.e. the end of the day). Similarly, consider a set of UGVs that need to map a building while taking into account their finite battery life.

In addition, DCOPs are single shot optimisation problems. For this reason, dynamism, which is a typical feature of most real world problems is not taken into account. Such dynamism is typically due to the agents moving and making decisions in the environment. Within a DCOP, this dynamism typically means that variables and constraint functions can appear or disappear at anytime. It also means that the constraint functions can change both in their outcomes and in the variables to which they are connected to

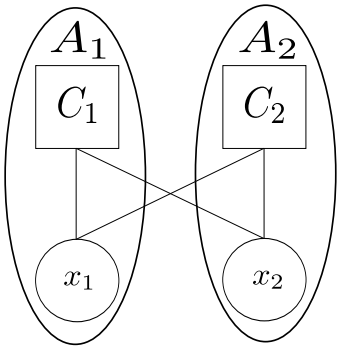


FIGURE 2.10: An example of a factor graph

reflect the variations in the different agents' interactions. For instance, consider again the target tracking problem, new targets will continuously appear and disappear while the UAVs are exploring the area. Similarly, in the target-search problem, the UAVs will continuously explore new places and their observations might overlap at different occasions.

However, despite dynamism cannot be encoded in a DCOP, it can be easily incorporated within a factor graph. In fact, the dynamism of most coordination problems is reflected in variations in the graph's topology. More specifically, it corresponds to the appearance or disappearance of new nodes, edges and connections. In other words, the topology of the factor graph will continuously change as new constraint functions and variables will be added and deleted. For instance, in target tracking, new functions will correpond to new targets to track, while new variables will corresponde to new UAVs. In target search, new variables will correspond again to new UAVs while new function nodes will correspond to new overlapping observations. To understand then how the new factor graph is computed and how it is decentralised between the agents, we will present in Chapter 3 a methodology to use the DCOP framework to address problems related to situational awareness.

In general, addressing dynamism is one of the key research challenges to address by research in coordination and in DCOPs. Indeed, despite it can be incorporated within a factor graph, a principled study of the ways in which dynamism can affect the solutions of a DCOP is still missing by literature. For this reason, dynamism is one of the key challenges to address in the prosecution of our work (see Chapter 8).

Nonetheless, DCOPs still remain a very promising framework for tackling coordination problems. One of the key reasons for this is one key property of DCOPs which we are going to exploit throughout this thesis to design new algorithms. More specifically, the algebraic structure of the global constraint function defined in Equation 2.1 is that of a commutative semi-ring, which is defined as follows:

Definition 2.2. A commutative semi-ring is a triple $\langle R, \oplus, \otimes \rangle$, where R is a non-empty set and $\oplus$ and $\otimes$ are two (abstract) binary associative and commutative operators over R, such that $\otimes$ distributes over $\oplus$. Furthermore, $\mathbf{0} \in R$ is an identity element such that $\forall x \in R : x \oplus \mathbf{0} = x$, and $\mathbf{1} \in R$ is an identity element such that $\forall x \in R : x \otimes \mathbf{1} = x$.

This property allows the objective function of a DCOP (Equation 2.1) to be disaggregated into the set of functions C (thanks to the operators $\oplus$ and $\otimes$). By exploiting it, as we will shortly see, GDL algorithms can optimally solve DCOPs by using very little computation (See Aji and McEliece (2000) for a more thorough description of the advantages of using the GDL), and as a consequence it satisfies the requirement of resource awareness which is crucial for defining coordination algorithm in our work (see Chapter 1). Additionally, GDL algorithms can be used to instantiate either optimal or bounded approximate algorithms and, as a consequence, they also satisfy the requirement of performance guarantees which is again crucial for our setting. The use of the GDL framework appears then as a natural choice for building our algorithms.

Having discussed the framework that we are going to use, we can now discuss DCOPs algorithms. In particular, we focus our attention on approximate, bounded approximate and optimal algorithms.

2.4.2 Coordination Algorithms for Totally Ordered Problems

This section discusses algorithms to solve canonical DCOPs. In essence, solving a DCOP consists in finding the optimum of its global constraint function in a decentralised fashion. We have already argued in the previous section how DCOPs are a good framework to represent coordination problems related to disaster response. For instance, they could be used to represent target search problems in which the agents coordinate to optimise a global constraint function representing the cumulative probability of the detecting the target(s). When this optimum cannot be obtained due to the agents' limited computational capabilities an approximate solution can be calculated instead, so as bounds on its quality. We discuss in this section these different categories of algorithms namely optimal, approximate and bounded approximated.

2.4.2.1 Optimal Algorithms

Four key algorithms have been proposed to solve DCOPs optimally, namely, OptAPO (Lesser et al., 2003), ADOPT (Modi et al., 2005), DPOP (Petcu and Faltings, 2005) and Action GDL (Vinyals et al., 2011). OptAPO uses a partially centralised approach in which mediator agents compute solutions for portions of the problem. ADOPT is based on a decentralised version of the branch and bound algorithm and works by

formulating the constraints of the problem as a graph, and then exchanging messages between the nodes of this graph (Modi et al., 2005). DPOP and Action GDL are again fully decentralised, and consist of a message passing procedure which, as we will shortly see, exploits the GDL framework to solve DCOPs.

The key disadvantage of these algorithms is that they all incur an exponential cost, either in the computation or in the communication required to provide optimal solutions. For instance, with OptApo, the agents might be required to perform calculations that grow exponentially with the size of the portion of the overall problem that they are responsible for. Hence requiring excessive computational resources to reach a timely solution. In ADOPT, the number of messages that agents exchange is exponential in the width of the tree, thus, exceeding by far the limited communication capacities of robotic agents such as UAVs. Similarly, as we will shortly see, both DPOP and Action GDL use a pre-processing step over the constraint graph representing a DCOP, which exponentially increases the size of the messages exchanged by the different agents (Petcu and Faltings, 2005; Vinyals et al., 2011).

Within our setting, we are interested in optimal algorithms since they meet the performance guarantees requirement discussed in the introduction. In more detail, we will use the Action GDL as one of the algorithms that we are going to extend to address multi objective, risk awareness and online learning in the following chapters. The reasons for this choice will be more clear in Section 2.4.4 when we will discuss the GDL framework.

2.4.2.2 Approximate Algorithms

Approximate approaches to solve DCOPs are typically decentralised sub-optimal algorithms which require limited communication and computation to produce a solution (Fitzpatrick and Meertens, 2003; Maheswaran et al., 2005). These approaches, despite being studied within the multi-agent system community, share many similarities with the explicit coordination approaches studied by the robotics community (as discussed in Section 2.3.3). Essentially they are both decentralised negotiation algorithms in which the agents share their utilities. However, research in multi-agent systems has focused on studying ways to improve the quality of the solution recovered by these algorithms, and consequently to improve the quality of the decision making of the agents that use them. Two key examples to illustrate this are the distributed stochastic algorithm (DSA) (Maheswaran et al., 2005) and the maximum gain messaging algorithm (MGM) (Fitzpatrick and Meertens, 2003). These algorithms work by having each agent make a decision based only on the communicated (or observed) decisions of those local neighbours that influence its utility. DSA is then a best response algorithm where, after the negotiation phase, an agent makes a decision depending on a pre-defined pdf. Similarly, MGM allows one agent to make a decision only after the utilities of all the neighbouring agents have

being collected. In addition, DSA has been shown to perform well in a very sophisticated disaster response simulated environment such as the Robocup Rescue (Chapman et al., 2009).

The main drawback of these algorithms is that they often converge to poor quality solutions. The reason for this is that agents do not explicitly communicate their utility for making a specific decision, but only communicate their preferred one (i.e. the decision that will maximise their own utility) based on the current preferred decision of their neighbours. In contrast, the max-sum algorithm which we are going to discuss in Section 2.4.4, allows the agents to communicate their utility and for this reason it has been shown to outperform algorithms such as MGM and DSA in a variety of settings, such as coordinating mobile sensors for monitoring spatial phenomena, patrolling and pursuit evasion (Stranders et al., 2009, 2010). For this reason, we will use max-sum as the algorithm to deploy on the coordination system for disaster response which we are going to present in Chapter 3.

2.4.2.3 Bounded Approximate Algorithms

Various frameworks have been proposed for calculating offline bounds on the performance of local approximate DCOP algorithms. The most common are the k-optimality and the t-distance frameworks (Pearce et al., 2005; Kiekintveld et al., 2010). In more detail, k-optimal and t-distance algorithms are based on coordinating the decisions of local groups (neighborhoods) of agents. Thus, given a DCOP, agents inside a neighbourhood coordinate to locally optimise their joint decision by considering any joint assignment that can improve their joint reward. The difference between k-optimal and t-distance algorithms is the criterion employed to generate neighbourhoods: k-optimal algorithms create neighbourhoods of a fixed size k, where k is the number of agents. In contrast, t-distance approaches create, for each agent, a neighbourhood that includes all other agents within a certain distance t in the constraint graph.

However, both of these frameworks assume that the topology of the constraint network is known beforehand and that it does not vary. This is highly unlikely in real world settings. For example, in the disaster response scenario presented in Chapter 1 and in any of the problems presented in Section 2.2, since the vehicles are moving the topology of the constraint network encoding their interactions are going to vary continuously. A similar framework was proposed by Vinyals et al. (2010), where a DCOP is solved by alternating divide and coordinate phases in which the problem is decomposed into subproblems that are then solved by the agent. The way the problem is decomposed can then be used to calculate bounds on the quality of the recovered solutions. Unfortunately, this approach suffers from the same drawbacks as k-optimality and t-distance since it assumes that the topology of the constraint network is static and known before-hand.

Conversely, online bounds have been computed for the max-sum algorithm by adding a pre-processing phase to the algorithm (Rogers et al., 2011). As we will see in more detail in the next section, this new phase is used to approximate a DCOP by deriving a new constraint graph which can be solved by max-sum whilst keeping track of the difference between the solution of the approximated and the original graph. In our work we will generalise the bounded max-sum algorithm to solve distributed constraint optimisation problems defined over multiple objectives. Indeed, as we will see in the Chapters 5 and 6 solving the former problems is extremely costly in terms of computation. Using a bounded approximate solution technique is then a reasonable compromise between the former complexity and the quality of the solutions recovered.

2.4.3 Coordination Algorithms for Partially Ordered Problems

This section discusses algorithms to solve coordination problems involving complex constraints functions. By complex, we mean functions defined over multiple parameters, such as multiple objectives or mean and variance characterising a pdf. In so doing, we present a set of algorithms related to the complex interaction requirement defined in Chapter 1. We focus here on the three types of interactions described in Chapter 1, multiple objective, various degrees of risk awareness and covering online learning. In contrast with the previous sections, however, we will not focus only on DCOP algorithms but also present algorithms pertaining to other fields such as constraint optimisation and machine learning. In so doing, we aim to present not only algorithms which are going to be used as benchmarks for our approaches, but also techniques which will constitute the foundation over which we will build such techniques.

2.4.3.1 Problems involving Multiple Objectives

Research in multi-objective optimisation studies problems where the agents (either human of software) have to discriminate between different *incommensurable* objectives. Here, the notion of incommensurability is used to emphasize the fact that the objectives are typically defined over different and independent criteria (see (Marler and Arora, 2004) for more details). For instance, consider the case of a UAV searching an area for a lost target, two incommensurable objectives could be minimising its battery consumption (i.e. measured in Joules) while maximising the area coverage (i.e. measured in m^2).

Unfortunately, multi-objective problems have been addressed by DCOP research to a very limited degree. Algorithms have been developed to solve DCOPs in which the agents' resources were modelled as new objectives of the problem (Bowring et al., 2006;

Kumar et al., 2009). Despite being the first work in this line of research, it just considers a specific case in which a new objective is defined over the agents' resources, and does not address general multi-objective problems. However, a vast literature exists on multi-objective optimisation. In general, due to the complexity of solving these problems, meta-heuristics such as genetic or evolutionary algorithms are typically considered (Veldhuizen and Lamont, 1998). Unfortunately, these algorithms do not fit our setting, since they are centralised and haven't been designed to coordinate agents deployed in real world application domains.

In contrast, recent research in constraint processing and optimisation has generalised constraint optimisation problems (COP) to the multi-objective setting (Rollón, 2008; Marinescu, 2009). Hence, considering that the COP and the DCOP frameworks share many similarities (i.e. a DCOP is simply a COP which has been extended to decentralised decision making problems by assigning variables to different agents), we shall use, for the sake of our exposition, the multi-objective COP (MO-COP) framework to describe multi-objective problems and their solutions. Formally, a MO-COP[2] is defined as the problem of simultaneously maximising k incommensurable *objective functions*, defined over a set $X = \{x_1, \ldots, x_n\}$ of N discrete variables, where each x_j takes values in a discrete domain D_j (the Cartesian product of all these domains $D = \times_{j \in \{1,\ldots,n\}} D_j$ is then the domain of the set X). Thus, a solution to a MO-COP is an assignment X^* of values to variables, such that:

$$X^* = \arg\max_X \mathbf{U}(X) = [U^1(X), \ldots, U^k(X)]^T \tag{2.2}$$

Here, each objective function U^i can be defined over a subset $X_i \subseteq X$ of the variables of the problem. However, for ease of exposition, we assume each function is defined over the same set of variables.

Now, since these objectives are often scalar values, many solution techniques work by reducing a multi-objective problem to a single objective problem that can then be solved using canonical optimisation techniques. The achieve this, a new objective function is defined as a weighted sum of each objective. The key challenge is then to define such weights in a way that can yield effective solutions. However, achieving this typically requires an exhaustive search of the space of all possible weights. Hence, these techniques are unlikely to lead to optimal or good quality solutions in a reasonable time, unless the solution space has some specific structure (e.g. it is convex) (Veldhuizen and Lamont, 1998).

Another possibility, is for the weights to encode the agents' preferences about the objectives (Talbi et al., 2008; Veldhuizen and Lamont, 1998). However, in such cases, as

[2]We use the same notation as for DCOPs due to the similarities between the two frameworks.

argued by literature, the motivation for using multi-objective optimisation falls short. Indeed, a multi-objective optimisation problem assumes that the objectives are either equally important or that such importance is unknown before solving the problem. If this importance was known, it would be possible to define a single objective function that weights each objective based on the agents' preferences and canonical solution techninques could then be used. More generally, if the agents have preferences towards the objectives, discriminating between each of them is not necessary anymore.

Hence, a key property of multi-objective optimisation is trading off between the different objectives. Hence, since the functions are incommensurable, multiple assignments might satisfy Equation 2.2. For example, consider three assignments X_1, X_2 and X_3, such that $\mathbf{U}(X_1) = [4, 5]$, $\mathbf{U}(X_2) = [4, 3]$, and $\mathbf{U}(X_3) = [6, 3]$. Clearly, $[4, 5]$ and $[6, 3]$ are larger than $[4, 3]$. However, $[4, 5]$ and $[6, 3]$ are not comparable. Thus, X_2 does not satisfy Equation 2.2. Indeed, Equation 2.2 involves the optimisation of sets of *partially-ordered* assignments. Thus, to characterise the optimal solutions of a multi-objective optimisation problem, we use the well known concept of *Pareto optimality*:

Definition 2.3 (Pareto Optimality (Pareto, 1906))**.** An assignment $X^* \in D$ is *Pareto optimal* iff there does not exist another assignment $X' \in D$, such that $\mathbf{U}(X') \geq \mathbf{U}(X^*)$, and $U^i(X') \succ U^i(X^*)$ for at least one objective function.

The utility vector $\mathbf{U}(X^*)$ corresponding to a Pareto optimal assignment X^* is referred to as a *non-dominated* vector. We define the notion of *non-dominance* as follows:

Definition 2.4 (Non-dominance)**.** A vector $\mathbf{U}(X^*) \in D$ is *non-dominated* iff there does not exist an assignment $X' \in D$, such that $\mathbf{U}(X') \geq \mathbf{U}(X^*)$, with at least one $U^i(X') > U^i(X^*)$. Otherwise, $\mathbf{U}(X^*)$ is said to be *dominated*.

Thus, since a multi-objective problem involves the optimisation over partially ordered assignments, its solution is a *set* of Pareto optimal assignments.

To solve MO-COPs using Pareto dominance, two extensions of well known single-objective algorithms have been proposed, namely the bucket elimination (Rollón and Larrosa, 2006) and the branch and bound algorithm (Rollón, 2008). The former is an optimal algorithm which can be considered as a centralised version of the DPOP algorithm discussed in Section 2.4.4. The latter is a bounded approximate algorithm which uses the well known concept of utopia point to compute the bounds on the quality of the solutions:

Definition 2.5. Utopia Point (Marler and Arora, 2004): The utopia point $\mathbf{V}^*$ of a multi-objective optimisation problem is given by:

$$\mathbf{V}^* = \left[\max_X U^1(X), \ldots, \max_X U^k(X)\right]$$

Put differently, the utopia point is the vector of values resulting from optimising each k optimisation problems independently. Clearly, given any Pareto optimal assignment X^*, the inequality $U(X^*) \leq \mathbf{V}^*$ holds. Unfortunately, the above mentioned algorithms are both centralised and, as a consequence, they do not meet the robustness requirement which is crucial for the type of problems that we wish to solve (1). However, given their multiple similarities with DCOP algorithms such as DPOP and ADOPT, they constitute a solid base over which to build the algorithms that we will present in Chapter 5.

2.4.3.2 Problems involving varying Risk Awareness

Researchers have recently started studying problems in which the outcome of the constraint functions of the DCOP is uncertain (Taylor et al., 2010; Jain et al., 2009; Léauté and Faltings, 2009, 2011). In a decision making context, this means that the outcome of the decisions of one or more agents is unknown. For instance, consider a setting in which UAVs are deployed to provide live aerial imagery of some of the most important areas of the scene of a disaster (see the scenario depicted in Chapter 1 for more details). Numerous factors make the outcome of the UAVs decisions uncertain. Some of the factors are related to the vehicles, such as the UAVs Global Positioning System (GPS) preventing them from knowing their position with precision or their battery capacity preventing them from knowing their remaining operation time with precision. Some of the factors are related to the environment, such as the wind, limiting the UAVs ability to fly and the visibility preventing the UAVs from collecting useful imagery. In such settings, Léauté and Faltings (2009, 2011) focused their attention on studying the sources of uncertainty and proposed an incomplete extention of DPOP to solve problems in which exogenous sources of uncertainty were modeled as uncontrollable random variables.

Now, the most natural way to solve coordination problems in which uncertainty is endemic is to have the agents coordinating their decision making considering the expected or mean value of the outcome of their decisions. However, in so doing, we would implicitly assume that the decision process is risk neutral. That is to say, that the agents do not take uncertainty into account and do not weight the risk associated with choosing to make an uncertain decision instead of a certain one. In fact, there exists, an extensive literature on decision making under uncertainty (see (Levy, 2006)) that indicates that preferences over outcomes should be determined by a utility function which encodes an agent's risk profile, defined as follows:

Definition 2.6 (Risk profile (Levy, 2006))**.** A risk profile indicates the willingness of an agent to expose itself to uncertain outcomes in the prospect of a higher utility.

To understand this consider again the scenario described in Chapter 1. In this setting, the risk profile modelling the attitude of the agents towards uncertain (and potentially

undesirable) outcomes can be of three different types. A risk averse profile defines agents more inclined to make certain but less rewarding decisions rather than highly rewarding (in expectation) but uncertain ones. For instance, the unmanned aerial vehicles (UAVs) will prefer exploring well known areas containing a lower but certain number of casualties instead of unknown locations, in which this number might be higher but unknown. A risk seeking profile then defines agents which prefer uncertain but more rewarding (in expectation) decisions over certain ones. For instance, by adopting this type of profile, the UAVs will choose to explore places expected to contain a higher number of casualties instead of places in which the number is known with precision but lower. Finally, a risk neutral profile defines agents indifferent between certain or uncertain decisions. Thus, by adopting this type of profile, the agents, will be indifferent to the uncertainty related to the locations that they are exploring. DCOPs extensions that model uncertain constraint functions have been proposed by literature.

Within a coordination problem, such as a DCOP, this utility function should then encode the risk profile of the whole collective of agents, that is to say, the collective's *joint preference* over a global outcome. In this vein, Atlas and Decker (2010) extended DCOPs to the setting in which constraint functions are stochastic. Although, they only considered discrete probabilities, they stated that the extention to continuous pdfs was straightforward. In addition, they proposed a new local approximate algorithm, the distributed neighbour exchange algorithm (DNE), which could be used to solve DCOPs involving risk neutral agent profiles, where the agents have no preference between a risk seeking or averse behaviour, and thus, simply maximise the expected outcome of the global function. However, DNE does not fit our settings, because it fails to provide any quality guarantees, nor does it handle risk seeking or averse behaviours which are strictly necessary in environments where uncertainty is endemic. In fact, most current DCOPs algorithms are only capable of maximising expected outcome and do not take into account the agents' joint preferences towards risk (Taylor et al., 2010; Jain et al., 2009; Léauté and Faltings, 2009, 2011). Hence, applying these algorithms to disaster response settings such as those discussed above, is likely to yield a poor performance in terms of the agents' decision making.

2.4.3.3 Problems involving Online Learning

There exists a vast amount of literature on online learning algorithms (Sutton and Barto, 1998). These algorithms are useful in stochastic and *a priori* unknown environments, in which agents are no longer faced with a one shot optimisation problem as encoded in a canonical DCOP. Rather, agents now have to coordinate to solve a sequence of "DCOP-like" problems in order to simultaneously reduce uncertainty about the local utility functions (*exploration*) and maximise the global utility (*exploitation*). In the context of disaster response a good setting in which online learning can be used is the

SLAM problem defined in Section 2.2. In this problem, robotics agents such as UAVs or UGVs are deployed to map an environment without any prior information about it. Moreover, their sensors may be noisy and imprecise. Given these constraints, the agents are forced to learn the outcome of each of their decisions during the actual operation. This means that they have to explore the outcome of their decisions, i.e. map the areas less likely to contain casualties and they have to exploit the best outcomes, i.e. map the areas which have been estimated to contain the highest number of casualties.

Now, focusing solely on exploration results in certainty about the agents' environment, but wastes resources by taking suboptimal actions. Similarly, consistently taking the joint action that is currently believed to be the best is also suboptimal because this belief might be incorrect. Hence, it is necessary to strike a balance between exploration and exploitation. This challenge is not satisfied yet by any existing DCOP algorithm. Indeed, either they make some assumption on the problem that simplifies the challenge of trading off between exploration and exploitation, or they address the challenge but in an inefficient fashion. For instance Taylor et al. (2010) and Jain et al. (2009) proposed a set of extensions to well known DCOP algorithms such as MGM and DSA to address DCOPs whose constraint functions are unknown. However, despite being *a priori* unknown, these functions are assumed to be non-stochastic. Hence, these algorithms address much simpler problems than the ones we want to address here. Furthermore, Léauté and Faltings (2009) proposed an extension of DPOP (referred as E-DPOP) in which the values of the constraint functions are influenced by exogenous sources of stochasticity, whose underlying probability distributions are unknown. Whilst a significant contribution to the field, all of this work divides exploration and exploitation into two distinct phases, a process that is known to make its performance dependent on the specific problem instance (Sutton and Barto, 1998). Moreover, as acknowledged by the authors, the E-DPOP algorithm is incomplete and as a result can perform arbitrarily poorly on general problem instances (Léauté and Faltings, 2011).

In contrast, the multi-armed bandit (MAB) community has addressed the trade off between exploration and exploitation in a principled fashion, but from a single agent perspective (Lai and Robbins, 1985; Auer et al., 2002; Vermorel and Mohri, 2005). In this context, MAB is a simple analytical tool for modeling decision making under uncertainty. More formally, a MAB is a slot machine with K arms, each of which delivers rewards drawn from an unknown distribution. The aim of the problem is to sequentially pull the arms so as to maximise the cumulative reward over a finite time horizon. In other words, the agent's goal is to choose which arms to pull so as to maximise expected cumulative reward over a finite time horizon T. In more detail, let P be a pulling policy (a sequence of pulls), and $i(t)$ denote arm chosen at time t by P, and $r_{i(t)}(t)$ the reward received by pulling that arm at time t. Then, we can formalise the agent's goal as follows:

$$P^* = \arg\max_{\mathbf{P}} \sum_{t=1}^{T} r_{i(t)}(t) \tag{2.3}$$

where $\mathbf{P}$ is the set of all possible policies. Clearly, if the reward distributions of each arm were known, the optimal policy would be to always pull the arm with the highest expected reward. This hypothetical scenario sets a performance benchmark known as *regret* against which to compare any policy. In particular, the regret $R_P(T)$ of a policy P after T time steps is the difference between the expected reward of the theoretical optimal policy and that obtained by P:

$$R_P(T) = \sum_{t=1}^{T} [\mu^* - \mu(i(t))] \tag{2.4}$$

where $\mu(i)$ is the *expected reward* of arm i and $\mu^* = \max_i \mu(i)$. Now, there exists a number of pulling policies for minimising regret, such as ε-first (Even-Dar et al., 2002), ε-greedy (Sutton and Barto, 1998) and upper confidence bound (UCB) (Auer et al., 2002). Whereas the former two are sensitive to the choice of the ε parameter, the latter provides optimal theoretical guarantees on the regret (the difference between its performance and that of the theoretical optimal solution) without any need for parameter tuning. Hence, UCB is one of the most widely used, since it is non-parametric and achieves asymptotically optimal regret. In more detail, UCB pulls each arm once at the beginning, then at each subsequent time step t, UCB selects arm $i^*(t)$ with the maximum upper confidence bound on the expected reward:

$$i^*(t) = \arg\max_{i \in [1,K]} \left[\hat{\mu}(i,t) + \sqrt{2 \ln t \frac{(u_{\max} - u_{\min})^2}{n(i,t)}} \right] \tag{2.5}$$

where $\hat{\mu}(i,t)$ is the sample mean of the rewards that arm i received until t, and $n(i,t)$ is the number of times UCB pulled arm i before time step t. UCB assumes the support of the reward function is bounded, i.e. $r_i(t) \in [u_{\min}, u_{\max}]$. The two terms in Equation 2.5 determine the trade off between exploration and exploitation. The larger the first, the more *exploitation* is favoured, since it is an estimate of the expected reward of arm i. The larger the second, the more *exploration* is favoured, since it represents the uncertainty in this estimate.

Now, MABs constitute a very interesting framework to represent online learning problems due to their simplicity and due to the fact that MAB algorithms present a set of theoretical properties that can be used to compute performance guarantees over the recovered solutions. The aim of this work, as we will present in Chapter 7, is then to merge this theory with the GDL framework in order to derive new algorithms to solve

DCOPs in which the agents need to learn from their environment. For this reason, in what follows we describe the GDL framework in detail.

2.4.4 The Generalised Distributive Law Framework

Thus far, four algorithms exist that exploit the GDL framework to solve DCOPs: DPOP, Action GDL, bounded max-sum and the max-sum algorithm. The aim of this section is to describe these algorithms, their similarities and differences. In so doing, we aim to describe the way in which the GDL framework can be used to derive optimisation algorithms, so that we can use it in the following chapters to derive new algorithms to solve complex coordination problems (as described in Chapter 1).

Now, depending on the complexity of the problem and on the quality of the solutions required, the GDL framework can be used in two different ways. First, it can be used to instantiate algorithms capable of providing optimal or bounded approximate solutions at the cost of a greater complexity, as discussed in the previous sections. Second, it can be used to instantiate incomplete or loopy approximate algorithms, capable of solving complex real world problems. In fact, extensive empirical evidence has shown that the use of such algorithms often yield high quality solutions even when applied to extremely complex problems (Aji and McEliece, 2000; Kschischang et al., 2001). In both cases, the optimisation problem is first encoded into a factor graph as discussed in Section 2.4.1 for the DCOP case (Kschischang et al., 2001).

Within our setting, we will be using the GDL framework in both ways. We will use the generic framework to derive algorithms capable of dealing with multiple objectives, uncertain and unknown outcomes. To do so, we will exploit a fundamental property of GDL algorithms (see Theorem 2.7), which is that they are optimal over acyclic factor graphs. Next, since, as described in Section 2.2, we will be dealing with a dynamic task assignment problem, we will then use a loopy approximate algorithm, namely max-sum, to coordinate UAVs in a disaster response scenario. The algorithm is essentially a GDL algorithm running over cyclic factor graphs. Specifically, we will use an extension of max-sum aimed at solving task assignment problems (Macarthur et al., 2011). In what follows, we first describe the way to instantiate the GDL to derive optimal or bounded approximate algorithms, specifically focusing on DCOP algorithms, and then we describe the specific max-sum algorithm for task assignment.

2.4.4.1 Message Passing Algorithms over Acyclic Graphs

Complete DCOP algorithms based on the GDL are DPOP, Action GDL and bounded max-sum. These algorithms proceed over three phases. The first is used to allow the algorithm to provide quality guarantees. The second constitutes the core of the GDL

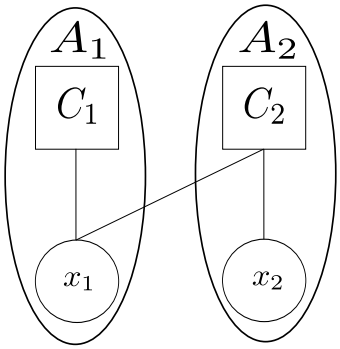


FIGURE 2.11: The spanning tree resulting from the factor graph in Figure 2.10

framework and consists of a message passing procedure to solve optimisation problems. The third is specifically related to DCOPs and is necessary to solve them in a consistent fashion. Each phase proceeds as follows:

Phase 1: Encoding the Objective Function as a Graph

This phase is independent from the type of optimisation problem that is being solved and is necessary to provide quality guarantees. To understand this, recall, from Section 2.4.1, that the global constraint function in Equation 2.1 is first encoded as a *factor graph*. If this graph is acyclic, then any GDL based algorithm is guaranteed to converge to an optimal solution (see Theorem 2.7). Thus, by using this property, it is possible to provide quality guarantees on the solutions recovered. The key idea is manipulating factor graphs until they become acyclic. Two different graphical approaches have been used in the DCOP literature.

The first approach is the one used for the bounded max-sum algorithm and is used to provide online bounded approximate solutions to the quality of the solutions recovered by the algorithm (Rogers et al., 2011). This approach consists of pruning the edges that have the least impact on the quality of the solutions. In so doing a bounded approximation can be derived. In more detail, it consists of computing a spanning tree of the original factor graph. To compute such a tree, edges are removed from the factor graph in a principled way. More specifically, here, the goal is to compute a variable assignment $\tilde{X}$ in the acyclic factor graph, such that:

$$C^* = \sum_{j=1}^{m} C_j(X_j^*) \leq \rho \sum_{j=1}^{m} C_j(\tilde{X}_j) \tag{2.6}$$

where X^* is the optimal solution of the cyclic factor graph (i.e. the "entire" problem), C^* the corresponding optimal value and ρ is the approximation ratio. To ensure this approximation ratio is as small as possible, the algorithm prunes those edges that have

the least impact on solution quality. The impact of an edge between variable x_i and constraint C_j is defined as its weight w_{ij}, defined as follows:

$$w_{ij} = \max_{X_j \setminus \{x_i\}} \left[\max_{x_i} C_j(X_j) - \min_{x_i} C_j(X_j) \right] \tag{2.7}$$

Once all the weights are computed, the GHS algorithm (Gallager et al., 1983) a decentralised message passing algorithm is used to compute a maximum spanning tree of the factor graph. The newly obtained acyclic factor graph is then used in the second phase, in which the max-sum algorithm is used to compute $\tilde{X}$, which is the optimal variable assignment to the "pruned" problem:

$$\tilde{X} = \arg\max_{X} \sum_{j=1}^{m} \min_{X_j^c} C_j(\tilde{X}_j) \tag{2.8}$$

where X_j^c is the set of variables that were eliminated from the scope of function C_j, corresponding to the edges that were pruned from the factor graph. The approximation ratio ρ is then given by:

$$\rho = 1 + (\tilde{C}_{\text{pruned}} + W - \tilde{C}_{\text{approx}})/\tilde{C}_{\text{approx}} \tag{2.9}$$

where W is the sum of the weights of the pruned edges, $\tilde{C}_{\text{pruned}} = \sum_j \min_{X_j^c} C_j(\tilde{X}_j)$ is the value of the solution of the pruned problem and $\tilde{C}_{\text{approx}} = \sum_j C_j(\tilde{X}_j)$ is the value of the solution of the overall approximated problem. Thus, an upper bound on the optimal solution can be computed as follows:

$$\tilde{C}_{\text{approx}} + W \geq C^* \tag{2.10}$$

Figure 2.11 shows an example of a spanning tree resulting from pruning the edge (C_1, x_2) from the factor graph depicted in Figure 2.10.

The second approach is used for DPOP and Action GDL to provide optimal solutions. Each algorithm uses a different graphical approach (Vinyals et al., 2011; Petcu and Faltings, 2005). However, the key idea in both cases is to transform the original factor graph into an acyclic one by clustering variables. This new factor graph is then solved optimally by running max-sum. In essence, the nodes of the original factor graph that represent variables are clustered together until the graph becomes acyclic. Each cluster of nodes is then replaced by a new node representing a meta-variable defined over the Cartesian product of the domains of the variables (Aji and McEliece, 2000; Kschischang et al., 2001). Formally, this approach is said to compute a junction tree out of the

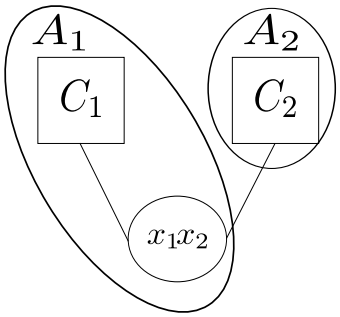


FIGURE 2.12: The junction tree resulting from the factor graph in Figure 2.10

Algorithm 1 The GDL algorithm for variable x_i. $\mathcal{M}(i)$ is the set of indices of neighbouring functions. $R_{j\to i}(x_i)$ is a message from function C_j computed in Algorithm 2

```
1: procedure GDL_VARIABLE(i)
2:     while stopping condition has not been met do
3:         for all j ∈ M(i) do                          ▷ For all adjacent functions
4:             if |M| = 1 or no messages received yet then
5:                 Send Q_{i→j}(x_i) = 1 to C_j
6:             else
7:                 Send Q_{i→j}(x_i) = ⊗_{k∈M(i)\j} R_{k→i}(x_i) to C_j
8:             end if
9:         end for
10:        Wait for new messages from all C_j : j ∈ M(i)
11:    end while
12:    return Z_i(x_i) = ⊗_{j∈M(i)} R_{j→i}(x_i)
13:    Assign x_i^* := arg max_{x_i} Z_i(x_i)
14: end procedure
```

original factor graph. The Action GDL algorithm uses such junction tree computation to solve DCOPs optimally. To this end, Figure 2.12 shows an example of a junction tree resulting from merging the two variables x_1 and x_2 of the factor graph in Figure 2.10. It should be noted, that by merging the two variables, agent A_2 loses control of variable x_2. The DPOP algorithm is then a specialisation of Action GDL, in which the junction tree computation is done by merging variables following a depth first procedure (Petcu and Faltings, 2005). However, by computing such junction trees, the optimal solution of the original problem is preserved, but the communication and computation overhead of the second phase are increased (Aji and McEliece, 2000).

Phase 2: Message-Passing

This phase constitutes the core of any algorithm based on the GDL framework[3]. In the context of DCOPs, the second phase computes the *marginal* global constraint function $Z_i(x_i)$ for each assignment to variable x_i:

[3]It corresponds to the UTIL-propagation phase in DPOP and Action-GDL.

$$Z_i(x_i) = \max_{X \setminus x_i} \sum_{j=1}^{m} C_j(X_j) \tag{2.11}$$

To understand the way in which this happens, we first discuss the generic framework and then we detail how it is instantiated for solving DCOPs. In more detail, Equation 2.11 is computed through message passing between the vertices of the graph constructed in phase 1. To perform this computation efficiently, GDL exploits the fact that the objective function is expressible using a *commutative semi-ring*, as discussed in Section 2.4.1. The message passing phase is completely decentralised and is implemented by Algorithms 1 and 2, which perform the computation associated with the variables and the functions respectively. Specifically, Algorithm 1 is executed by agent $F(x_i)$ that controls variable x_i. The algorithm computes the messages $Q_{i \to j}(x_i)$ that x_i sends to each of its neighbours C_j with $j \in \mathcal{M}(i)$ (line 3 of the algorithm). Similarly, Algorithm 2 can be executed by any of the agents whose variables are a parameter of C_j. The algorithm computes the messages $R_{j \to i}(x_i)$ that constraint C_j sends to each of its neighbours x_i with $i \in \mathcal{N}(j)$ (line 18 of the algorithm).

It should be noted that each type of message is defined in terms of the other one (lines 7 of Algorithm 1 and 19 of Algorithm 2). Indeed $R_{j \to i}(x_i)$ messages are defined considering the $Q_{i \to j}(x_i)$ messages that have been received (lines 18 - 20 of algorithm 2). Similarly, $Q_{i \to j}(x_i)$ messages are defined by aggregating all the $R_{j \to i}(x_i)$ messages that have been received. At the beginning of the computation, or when communication is limited, no messages are received. In this case, the algorithms send constant values (lines 5 and 19 of algorithms 1 and 2 respectively). Hence, GDL algorithms work as distributed protocols. More specifically, the computation starts by having the agent send constant values, then the agents iteratively share messages until a termination condition is met (lines 2 and 16 of Algorithms 1 and 2 respectively). This condition depends on the topology of the factor graph and will be discussed shortly. Finally, it should be noted that, for ease of exposition, the formulation of these algorithms is only given for acyclic factor graphs and assuming perfect communication between the agents. We will modify this formulation when describing max-sum for task assignment.

Now, the following theorem is a fundamental property of the GDL message passing algorithm:

Theorem 2.7. *If the factor graph is acyclic and the stopping criterion is chosen such that the algorithm is run for a number of iterations equal to the diameter of the factor graph, the following equation holds for each variable $x_i \in X$:*

$$Z_i(x_i) = \bigoplus_{X \setminus x_i} \bigotimes_{j=1}^{m} C_j(X_j) \tag{2.12}$$

Algorithm 2 The GDL algorithm for function C_j. $\mathcal{N}(j)$ is the set of indices of neighbouring variables. $Q_{i\to j}(x_i)$ is a message from variable x_i computed in Algorithm 1.

15: **procedure** GDL_FUNCTION(j)
16: **while** stopping condition has not been met **do**
17: Wait for new messages from all $x_i : i \in \mathcal{N}(j)$
18: **for all** $i \in \mathcal{N}(j)$ **do** ▷ For all adjacent variables
19: Send to x_i message $R_{j\to i}(x_i) =$

$$\bigoplus_{X_j \setminus \{x_i\}} \left(C_j(X_j) \otimes \bigotimes_{k \in \mathcal{N}(j)\setminus i} Q_{k\to j}(x_k) \right)$$

20: **end for**
21: **end while**
22: **end procedure**

For proof of this theorem, see (MacKay, 2003) (Chapter 26) and (Aji and McEliece, 2000) (Theorem 3.1).

For the sake of solving DCOPs, the GDL message passing phase is instantiated with the max-sum commutative semi-ring $\langle \mathbb{R}, \max, + \rangle$ and the resulting algorithm is often referred as the max-sum algorithm (Rogers et al., 2011; Frey and Dueck, 2007). By so doing, it is possible to obtain the *marginal* global constraint equivalent to Equation 2.11. The following example illustrates the behaviour of the algorithm:

Example 2.1. *This example illustrates the behaviour of the message passing phase, by running it over the acyclic factor graph depicted in Figure 2.11. We assume that the variables are defined over a binary domain, x_1, $x_2 \in \{0, 1\}$. The constraint functions C_1 and C_2 are defined as follows:*

x_1	x_2	$C_1(x_1)$	$C_2(x_1, x_2)$	$C(x_1, x_2)$
0	*0*	*5*	*4*	*9*
0	*1*	*5*	*10*	*15*
1	*0*	*8*	*7*	*15*
1	*1*	*8*	*2*	*10*

The stopping condition for the algorithm (lines 2 *and* 12 *of Algorithms 1 and 2) is a constant representing the number of iterations necessary for the values of the constraints to propagate throughout the graph. For acyclic graphs this number is typically equal to twice the diameter of the factor graph (MacKay, 2003). The computation starts from the variables (lines* 4 *and* 6 *of Algorithm 1):*

***Iteration 1**: Variables x_1 and x_2 compute the following messages:*

$Q_{1\to 1}(x_1) =$

x_1	C
0	0
1	0

$Q_{1\to 2}(x_1) =$

x_1	C
0	0
1	0

$Q_{2\to 2}(x_2) =$

x_2	C
0	0
1	0

***Iteration 2**: Constraints C_1 and C_2 receive the messages from iteration 1 and compute the new messages:*

$$R_{1\to1}(x_1) = \begin{array}{|c|c|}\hline x_1 & C \\ \hline 0 & 5 \\ 1 & 8 \\ \hline\end{array} \quad R_{2\to1}(x_1) = \begin{array}{|c|c|}\hline x_1 & C \\ \hline 0 & 10 \\ 1 & 7 \\ \hline\end{array} \quad R_{2\to2}(x_2) = \begin{array}{|c|c|}\hline x_2 & C \\ \hline 0 & 7 \\ 1 & 10 \\ \hline\end{array}$$

***Iteration 3**: Variables x_1 and x_2 receive the new messages and send them to the corresponding constraints:*

$$Q_{1\to1}(x_1) = \begin{array}{|c|c|}\hline x_1 & C \\ \hline 0 & 10 \\ 1 & 7 \\ \hline\end{array} \quad Q_{1\to2}(x_1) = \begin{array}{|c|c|}\hline x_1 & C \\ \hline 0 & 5 \\ 1 & 8 \\ \hline\end{array} \quad Q_{2\to2}(x_2) = \begin{array}{|c|c|}\hline x_2 & C \\ \hline 0 & 0 \\ 1 & 0 \\ \hline\end{array}$$

***Iteration 4**: Function C_1 and C_2 receive the new messages and send them to the corresponding constraints:*

$$R_{1\to1}(x_1) = \begin{array}{|c|c|}\hline x_1 & C \\ \hline 0 & 5 \\ 1 & 8 \\ \hline\end{array} \quad R_{2\to1}(x_1) = \begin{array}{|c|c|}\hline x_1 & C \\ \hline 0 & 10 \\ 1 & 7 \\ \hline\end{array} \quad R_{2\to2}(x_2) = \begin{array}{|c|c|}\hline x_2 & C \\ \hline 0 & 15 \\ 1 & 15 \\ \hline\end{array}$$

At the end of iteration 4, each variable has received the number of messages necessary to compute the maximum marginal function (Equation 2.11). This happens as follows:

$$x_1^* = \max\left\{\begin{array}{|c|c|}\hline x_1 & C \\ \hline 0 & 15 \\ 1 & 15 \\ \hline\end{array}\right\}$$
$$= \{0, 1\}$$

$$x_2^* = \max\left\{\begin{array}{|c|c|}\hline x_2 & C \\ \hline 0 & 15 \\ 1 & 15 \\ \hline\end{array}\right\}$$
$$= \{0, 1\}$$

Thus the second phase is capable of recovering the two optimal solutions of the problem. However, it might happen that the variables end up picking a sub-optimal assignment if they do not consistently decide which solution to select. To achieve this, a value propagation phase is used, which constitutes the third phase of these algorithms and which is going to be detailed next.

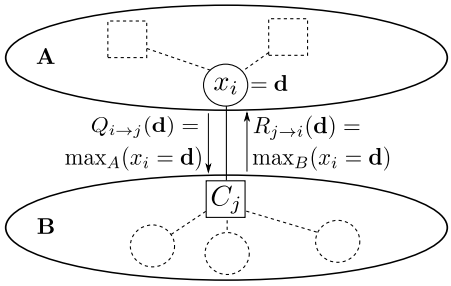


FIGURE 2.13: A graphical explanation of the GDL message passing procedure.

In essence, this second phase provides an efficient local message-passing procedure to compute the dependency of the global constraint function defined by Equation 2.1 on each of the variables in X simultaneously. This efficiency is due to the fact that each max-sum message propagates more information if compared with other message passing algorithms, such as those described in Section 2.3, which just share the agents' utilities for making a specific collective decision. Here, as illustrated by Figure 2.13, each message conveys the maximum aggregate of the global constraint function possible over the two components of the factor graph formed by removing the edge between C_j and x_i, for each value d of the domain D_i of x_i. In other words, the size of each message is linear in the domain of variable x_j.

Phase 3: Obtaining the Solution:

After the message passing phase is complete, Equation 2.1 is used to find the optimal variable assignment x_i^* for each variable $x_i \in X$ by setting $x_i^* = \arg\max_{x_i} Z_i(x_i)$. Note that this is only guaranteed to yield a consistent solution if the optimal solution is unique. Otherwise, variables can select assignments that correspond to different optimal solutions, resulting in suboptimality. To prevent this, a value propagation phase can be used to select an optimal solution using additional message passing (Petcu and Faltings, 2005).

In more detail, this value-propagation phase is again fully decentralised, and proceeds by passing messages between the variables and the functions in the acyclic factor graph. First, the variable x_i with the lowest index is chosen as the *root* of the tree, and is responsible for initiating the value propagation phase by selecting an optimal assignment $x_i = d_i^*$. The variable then sends value-propagation messages ($x_i = d_i^*$) to all the function nodes to which the variable is connected. The behaviour of all the other nodes in the graph will then depend on their type. More specifically:

Function nodes: Upon receiving a message ($x_i = d_i^*$) from variable x_i, the constraint function C_j computes the set $D^*_{X_j}$ of local optimal assignments for variables $X_j \setminus$

$\{x_i\}$, conditioned on $x_i = d_i^*$:

$$D^*_{X_j} = \left\{ d^*_{X_j} \middle| d^*_{X_j} = \underset{d_{X_j} \in (D_{x_j} \setminus D_i) \cup \{x_i = d_i^*\}}{\arg\max} \left[C_j(d_{X_j}) + \sum_{k \in \mathcal{M}(i) \setminus j} Q_{k \to j}(d_k) \right] \right\}$$

After computing set $D^*_{X_j}$, value propagation selects one assignment $d^*_{X_j} \in D^*_{X_j}$ and sends the message $(x_k = d_k^*)$ to every variable x_k $(k \neq j)$, where d_k^* is the element of $d^*_{X_j}$ corresponding to x_k.

Variable nodes: For each non-root variable x_i, once it receives a message $(x_i = d_i^*)$ from a function C_j, it sets its value to d_i^* and propagates the message $(x_i = d_i^*)$ to all the function nodes C_k, $k \neq j$.

Note that, during value propagation, a single message is sent across each link in the factor graph. Thus, the algorithm terminates once each non-root variable has received a value-propagation message.

The following example follows from Example 2.1 and illustrates the behaviour of value-propagation:

Example 2.2. ***Iteration 1****: Variable x_1 has the smallest index, and consequently it is elected as the root of the tree. It selects one best assignment $(x_1 = 0)$ and sends it to its neighbouring constraints C_1 and C_2.*

Iteration 2*: C_1 and C_2 receive the message from x_1. C_1 is only connected to x_1 and as a consequence does not compute anything. In contrast, C_2 calculates $D^*_{X_2} = \{x_1 = 0, x_2 = 1\}$ and sends $(x_2 = 1)$ to x_2.*

Iteration 3*: x_2 receives the message from C_2 and selects its value. Not having any other neighbours different than C_2 the computation ends.*

Thus, the use of the value propagation phase allows complete algorithms that exploit the GDL to break the ties of the values of the solutions recovered during the message passing phase. This also holds when solving DCOPs.

Unfortunately, complete algorithms are still not suitable to address realistic coordination problems (Section 2.2) and to be deployed within a real architecture (Section 2.3.3) since they require exponential computation and communication. From a theoretical perspective, this is to be expected, since we are now trying to optimally solve problems which have been demonstrated to be NP-hard (Modi et al., 2005). Nonetheless, these algorithms have been shown to be reasonably efficient and to scale more than other

optimal DCOP algorithms such as ADOPT and OptAPO (Petcu and Faltings, 2005; Vinyals et al., 2011). In addition, reducing the complexity of the algorithms, is one of our ongoing challenges and will be discussed in our concluding chapter (Chapter 8).

Fortunately, the GDL framework can be used to instantiate incomplete algorithms which fits better the limited computation and communication resources that characterise robotic agents. In what follows, having discussed the generic framework, we focus specifically on the incomplete max-sum algorithm which we are going to deploy in this thesis. More specifically, we discuss a specialisation of the max-sum algorithm that has been shown to greatly reduce the algorithm's computation and communication requirements in generic task assignment domains, such as the one we consider in this work.

2.4.4.2 A Message Passing Algorithm over Cyclic Graphs: Loopy Max-Sum

The max sum algorithm is a GDL algorithm instantiated with the max-sum commutative semi-ring $\langle \mathbb{R}, \max, + \rangle$. As with any GDL algorithm, it runs over a factor graph and consists of two message passing procedures based on Algorithms 1 and 2, which are used to compute Equation 2.11, that is to say, the maximum marginal of Equation 2.1. Now, considering the limited amount of communication and computation available to real robotic agents, quality guarantees cannot be provided anymore. For this reason, phase 1 is removed, and in so doing the algorithm becomes approximate, as discussed in Section 2.4.2.2. This means that the algorithm is now used over cyclic factor graphs. Nonetheless, extensive empirical evidence has shown that the former algorithms can generate good approximate solutions when applied to such cyclic constraint graphs. In fact, the incomplete version of max-sum has been shown to perform efficiently in a number of coordination problems, such as coordinating mobile sensors for patrolling and pursuit evasion scenarios and to monitor spatial phenomena, such as temperature or radiation (Stranders et al., 2009, 2010). In addition, this effectiveness, was also demonstrated in optimisation problems other than DCOPs, such as approximate inference on Bayesian networks (Murphy et al., 1999), iterative decoding of practical error correcting codes (MacKay, 1999), clustering of large datasets (Frey and Dueck, 2007), and solving large scale K-SAT problems involving thousands of variables (Mezard et al., 2002).

To achieve such performance, Algorithms 1 and 2 need to be slightly modified. Specifically, the stopping condition (lines 2 and 16) is eliminated so that messages are computed given a pre-defined schedule (e.g. periodically or in response of the receipt of a message) independent from the algorithm. Moreover, messages may be delayed or lost. Hence, new messages are computed given the last messages received by each node. This means that lines 10 and 17 are no longer necessary. In addition, only the received messages are used to calculate the new messages (lines 5, 6 and 19 in Algorithms 1 and 2). The same applies for computing Equation 2.11. Hence, lines 12 and 13 of Algorithm 1 are

Algorithm 3 The incomplete max-sum algorithm for variable x_i. $\mathcal{M}(i)$ is the set of indices of neighbouring functions. $R_{j\to i}(x_i)$ is a message from function C_j computed in Algorithm 4

1: **procedure** MS_VARIABLE(i)
2: **for all** $j \in \mathcal{M}(i)$ **do** ▷ For all adjacent functions
3: **if** $|\mathcal{M}| = 1$ or no messages received yet **then**
4: Send $Q_{i\to j}(x_i) = 0$ to C_j
5: **else**
6: Send $Q_{i\to j}(x_i) = \alpha_{ij} + \sum_{k\in\mathcal{M}(i)\setminus j} R_{k\to i}(x_i)$ to C_j ▷ For all k such that $R_{k\to i}(x_i) \neq \emptyset$
7: **end if**
8: **end for**
9: **end procedure**

no longer necessary, since this operation is now separated from the computation of the new messages. These changes lead to Algorithms 3 and 4, which illustrate the new procedures.

As shown in line 6 of Algorithm 3, a constant α_{ji} is added to each new message such that $\sum_{x_i} Q_{i\to j}(x_i) = 0$. This normalisation is used to prevent new messages from growing arbitrarily large when the factor graph contains cycles (see (Rogers et al., 2011) for more details). Phase 3 is removed to prevent further communication. To allow the agents to select a unique solution consistently, very small random values are added to the local constraint functions in C (Rogers et al., 2011) before commencing phase 2. Most importantly, the aggregation of the messages flowing into each variable now represents an approximate solution to the maximisation problem defined by Equation 2.1 with respect to x_i (i.e. it represents an approximation of the marginal function defined in Equation 2.11).

Now, within a task assignment setting such as the one we will address in this work, the UAVs' decisions consist of tasks to complete. Thus, each domain D_i is redefined as $D_i = \mathcal{T}_i$ where $\mathcal{T}_i \in \mathcal{T}$ is the set of tasks that agent i is aware of, out of all the tasks $T_j \in \mathcal{T}$. The constraints then represent the utilities of each possible assignment of one or more UAV to a task T_j. Within this setting, considering that messages are linear in the domains of the variables, when the number of tasks is very large the use of the canonical max-sum algorithm is likely to incur a significant increase in the amount of communication required. Moreover, considering that computing messages from functions to variables is exponential in the number of variables. Hence, when the number of tasks is very large, computing max-sum messages can become prohibitively expensive ($O(|\mathcal{T}|^{\mathcal{N}(j)})$ in the worst case for function to variable messages) and their size also becomes too large.

To address this shortcoming, max-sum has been modified to reduce the size and the complexity of computing the messages within each node (Macarthur et al., 2011). In

Algorithm 4 The max-sum algorithm for function C_j. $\mathcal{N}(j)$ is the set of indices of neighbouring variables. $Q_{i\to j}(x_i)$ is a message from variable x_i computed in Algorithm 3.

10: **procedure** MS_FUNCTION(j)
11: **for all** $i \in \mathcal{N}(j)$ **do** ▷ For all adjacent variables
12: Send to x_i message $R_{j\to i}(x_i) =$

$$\max_{X_j \setminus \{x_i\}} \left(C_j(X_j) + \sum_{k \in \mathcal{N}(j) \setminus i} Q_{k\to j}(x_k) \right) \quad \triangleright \text{ For all } k \text{ such that } Q_{k\to i}(x_k) \neq \emptyset$$

13: **end for**
14: **end procedure**

particular, the key idea is for each message passed between C_j and x_i to convey the maximum aggregate utility only for x_i being assigned to T_j and for x_i being assigned to some other task (i.e. max-sum messages become bi-valued vectors and the computation of function to variable messages is reduced to $O(2^{\mathcal{N}(j)})$). Therefore, whereas in standard max-sum the messages are defined as functions of the variables, in the modified version, each message is defined over an indicator function $\mathcal{I}_i^j(x_i)$ such that $\mathcal{I}_i^j(x_i) = 1$ when $x_i = T_j$ and $\mathcal{I}_i^j(x_i) = 0$ when $x_i \neq T_j$. The way in which the messages are computed then changes as follows:

Messages from variable x_i to function C_j (lines 3 - 6 of Algorithm 3):

$$Q_{i\to j}(\mathcal{I}_i^j(x_i)) = \begin{cases} \max_{k \in \mathcal{M}(i)\setminus\{j\}} \left(R_{k\to i}(1) + \sum_{w \in \mathcal{M}(i)\setminus\{k,j\}} R_{w\to i}(0) \right) & (x_i \neq T_j) \\ \sum_{k \in \mathcal{M}(i)\setminus\{j\}} R_{k\to i}(0) & (x_i = T_j) \end{cases}$$

Messages from function C_j to variable x_i lines 12, 13 of Algorithm 4:

$$R_{i\to j}(\mathcal{I}_i^j(x_i)) = \max_{X_j \setminus \{x_i\}} \left(C_j(X_j) + \sum_{k \in \mathcal{N}(j)\setminus\{i\}} Q_{k\to j}(\mathcal{I}_k^j(x_k)) \right)$$

At any time, each agent i can then compute its best assignment x_i^* as the sum of the messages flowing into x_i:

$$x_i^* = \arg\max_{x_i} \sum_{j \in \mathcal{M}(i)} R_{j\to i}(\mathcal{I}_i^j(x_i)) \tag{2.13}$$

Here, Equation 2.13 specialises Equation 2.11 to the task assignment setting.

Example 2.3 illustrates the approximate behaviour of the algorithm:

Example 2.3. *This example illustrates the behaviour of the loopy max-sum algorithm, by running it over the factor graph depicted in Figure 2.10. As indicated in the figure, two agents are defined which can attend both the tasks ($x_1, x_2 \in \{T_1, T_2\}$). The two tasks*

T_1 and T_2 are valued using two utility functions C_1 and C_2. The utilities are defined as follows:

x_1	x_2	$C_1(\mathcal{I}_1^1(x_1), \mathcal{I}_2^1(x_2))$	$C_2(\mathcal{I}_1^2(x_1), \mathcal{I}_2^2(x_2))$	$C(\mathcal{I}(x_1), \mathcal{I}(x_2))$
0	*0*	*5*	*4*	*9*
0	*1*	*3*	*5*	*8*
1	*0*	*6*	*5*	*11*
1	*1*	*4*	*9*	*13*

In this setting, in contrast to the one presented in Example 2.2, the algorithm is stopped after a certain number of seconds have passed. This stopping condition, as we will see in Chapter 3, is the one that is typically used to solve real world problems. In addition, it should be noted that for the sakes of a clearer exposition, we do not consider the normalisation constant α_{ji} in the computation of the messages. The computation starts from the variables (lines 4 and 6 of Algorithm 3):

***Iteration 1**: Variables x_1 and x_2 compute the following messages:*

$$Q_{1\to1}(x_1) = \begin{array}{|c|c|}\hline \mathcal{I}(x_1) & C \\ \hline 0 & 0 \\ 1 & 0 \\ \hline \end{array} \qquad Q_{1\to2}(x_1) = \begin{array}{|c|c|}\hline \mathcal{I}(x_1) & C \\ \hline 0 & 0 \\ 1 & 0 \\ \hline \end{array}$$

$$Q_{2\to1}(x_2) = \begin{array}{|c|c|}\hline \mathcal{I}(x_2) & C \\ \hline 0 & 0 \\ 1 & 0 \\ \hline \end{array} \qquad Q_{2\to2}(x_2) = \begin{array}{|c|c|}\hline \mathcal{I}(x_2) & C \\ \hline 0 & 0 \\ 1 & 0 \\ \hline \end{array}$$

***Iteration 2**: Constraints C_1 and C_2 receive the messages from iteration 1 and compute the new messages:*

$$R_{1\to1}(x_1) = \begin{array}{|c|c|}\hline \mathcal{I}(x_1) & C \\ \hline 0 & 6 \\ 1 & 5 \\ \hline \end{array} \qquad R_{1\to2}(x_2) = \begin{array}{|c|c|}\hline \mathcal{I}(x_2) & C \\ \hline 0 & 4 \\ 1 & 6 \\ \hline \end{array}$$

$$R_{2\to1}(x_1) = \begin{array}{|c|c|}\hline \mathcal{I}(x_1) & C \\ \hline 0 & 5 \\ 1 & 9 \\ \hline \end{array} \qquad R_{2\to2}(x_2) = \begin{array}{|c|c|}\hline \mathcal{I}(x_2) & C \\ \hline 0 & 5 \\ 1 & 9 \\ \hline \end{array}$$

***Iteration 3**: Variables x_1 and x_2 receive the new messages and send them to the corresponding constraints:*

$$Q_{1\to1}(x_1) = \begin{array}{|c|c|}\hline \mathcal{I}(x_1) & C \\ \hline 0 & 5 \\ 1 & 9 \\ \hline \end{array} \qquad Q_{1\to2}(x_1) = \begin{array}{|c|c|}\hline \mathcal{I}(x_1) & C \\ \hline 0 & 6 \\ 1 & 5 \\ \hline \end{array}$$

$$Q_{2\to1}(x_2) = \begin{array}{|c|c|}\hline \mathcal{I}(x_2) & C \\ \hline 0 & 5 \\ 1 & 9 \\ \hline \end{array} \qquad Q_{2\to2}(x_2) = \begin{array}{|c|c|}\hline \mathcal{I}(x_2) & C \\ \hline 0 & 4 \\ 1 & 6 \\ \hline \end{array}$$

Iteration 4*: Function C_1 and C_2 receive the new messages and send them to the corresponding constraints:*

$$R_{1\to1}(x_1) = \begin{array}{|c|c|}\hline \mathcal{I}(x_1) & C \\ \hline 0 & 15 \\ 1 & 14 \\ \hline \end{array} \qquad R_{1\to2}(x_2) = \begin{array}{|c|c|}\hline \mathcal{I}(x_2) & C \\ \hline 0 & 14 \\ 1 & 12 \\ \hline \end{array}$$

$$R_{2\to1}(x_1) = \begin{array}{|c|c|}\hline \mathcal{I}(x_1) & C \\ \hline T_1 & 19 \\ T_2 & 14 \\ \hline \end{array} \qquad R_{2\to2}(x_2) = \begin{array}{|c|c|}\hline \mathcal{I}(x_2) & C \\ \hline T_1 & 15 \\ T_2 & 18 \\ \hline \end{array}$$

At the end of iteration 4, each variable has received the number of messages necessary to compute the maximum marginal function (Equation 2.13). This happens as follows:

$$x_1^* = \max \left\{ \begin{array}{|c|c|}\hline x_1 & C \\ \hline T_1 & 34 \\ T_2 & 28 \\ \hline \end{array} \right\}$$
$$= \{0\}$$

$$x_2^* = \max \left\{ \begin{array}{|c|c|}\hline x_2 & C \\ \hline T_1 & 29 \\ T_2 & 30 \\ \hline \end{array} \right\}$$
$$= \{1\}$$

Thus the second phase recovers an approximate solution $\{x_1 = T_1, x_2 = T_2\}$ which is different than the optimal one $\{x_1 = T_2, x_2 = T_2\}$.

This version of max-sum is the one we are going to use in our practical contributions. However, to deploy it effectively a number of other issues need to be considered, such as the way to address the dynamism (new tasks or agents appearing / disappearing) and the way to effectively assign the nodes to the available agents. We will discuss these issues in detail in Chapter 3.

2.5 Summary

This chapter reviewed the literature related to our work. Specifically, we presented both similar research and research which will constitute the foundation over which we will build our algorithms. In more detail, Section 2.1 introduced UAVs, the specific type of robotic agent over which we are going to deploy our coordination system (see Chapter 3), and justified our choice to use custom made off-the-shelf hexacopters for our tests since they are less expensive and more practical for doing research. Section 2.2 then introduced the key research problems related to achieve situational awareness in disaster response scenarios. In our work, we will use these problems as an inspiration for designing our coordination algorithms (Chapters 4, 5, 6 and 7). In Section 2.3, we then presented practical decentralised coordination algorithms. More specifically, we presented various mechanisms that have been deployed in real vehicles for situational awareness missions and discussed the way we are going to use them in Chapter 3, for deploying the max-sum algorithm in our work.

Unfortunately, the former practical approaches, despite being applied to real world problems, are essentially distributed negotiation protocols, in which the agents share their decisions for a pre-defined amount of time until making a decision. For this reason, the level of performance that they achieve is typically very low. Moreover, they are not capable of addressing complex agent interactions as required in our work (as described in Chapter 1). In contrast, there exists a large body of literature on theoretical coordination approaches which has focused on more sophisticated interactions. These approaches are presented in Section 2.4. More specifically, we presented the DCOP framework, which we will use to define coordination problems involving complex interactions (see Chapter 4). In Section 2.4.2, we then presented optimal and bounded approximate algorithms for solving DCOPs. Next, we presented algorithms for solving problems involving complex interactions such as multiple objectives, uncertainty over the agents decisions and online learning (Section 2.4.3). In section 2.4.4, finally, we described the GDL framework and showed how it can be used to derive effective optimal, bounded approximate and simply approximate algorithms such as DPOP, Action GDL and max-sum. These reasons motivated our choice to use this framework to derive algorithms capable of solving coordination problems involving complex interactions (see Chapter 4).

Now, all the algorithms described above suffer from two main drawbacks which, as discussed in Chapter 1, motivate the research that we are going to present in the following chapters. The first reason is a practical one. We discussed in Section 2.4.4, how max-sum is a very good candidate to be deployed to coordinate real robotic agents. However, thus far the effectiveness of the algorithm has been demonstrated only in simulation. Thus, the efficiency of algorithms exploiting the GDL for coordinating the behaviour of multiple agents such as UAVs or UGVs still needs to be demonstrated in practice. This

is the focus of the next chapter, in which we present a case study on the deployment of max-sum for disaster response and discuss the various ways in which the algorithm can be efficiently deployed for this type of problems.

The second reason is a theoretical one. In Section 2.4.3 we presented three key examples of algorithms for solving problems involving complex agents' interactions. However, these approaches are still not usable within our setting. Indeed, either they address these problems from a single agent perspective, either they are centralised or they are not capable of providing any quality guarantees. For this reason, in Chapters 4, 5, 6 and 7 we go back to theory and derive new problems and algorithms to represent and solve coordination problems involving complex agent interactions.

Chapter 3

Application of the Max-Sum Algorithm for Disaster Response Scenarios

This chapter describes the main practical contributions of this thesis. The focus is to present a study of the deployment of the max-sum algorithm to achieve situational awareness in disaster response scenarios. We chose to use max-sum since, as discussed in Chapters 1 and 2, it is a good candidate to coordinate robotic agents such as UAVs or UGVs since it provides effective solutions whilst preserving the agents' limited computational and communication resources. However, despite this potential, thus far, max-sum has only been tested in simulation. As a consequence, max-sum's robustness and flexibility for real applications have not been tested in practice.

To address this shortcoming, this chapter provides two key contributions. The first is to describe a potential system whereby we deploy max-sum to coordinate a team of unmanned aerial vehicles to provide live aerial imagery to the first responders operating in the area of a disaster. However, as discussed in Chapter 1, max-sum's performance greatly depends on the way it is applied to a problem. For this reason, users less familiar with max-sum, might have trouble deploying the algorithm efficiently. This performance is affected by both the way in which a generic problem is encoded to form the input of the algorithm, and the way the algorithm is decentralised between the available sources of computation. In addition, whereas there exists a variety of ways to apply the algorithm in the literature, a general framework that discusses and analyses these issues is absent. Hence, the second contribution of this chapter (Section 3.1) introduces a methodology that a developer can use to deploy max-sum for problems related to situational awareness. This methodology is a set of general rules that unify

the different ways in which the algorithm can be applied to these problems and analyses their advantages and disadvantages.

Section 3.2 then presents a case study on disaster response. Specifically, we consider a scenario similar to the one presented in Chapter 1 in which a team of UAVs interacts with first responders to provide real-time live aerial imagery of specific sites of the area of a disaster, such as damaged buildings or flooded streets. Each first responder is provided with a personal digital assistant (PDA) that he uses to submit imagery requests in the form of tasks and to collect such imagery as soon as it becomes available. Each PDA uses a touch screen interface which allows the first responder to request imagery tasks about any area in his proximity and to specify the task's importance. The aim of the UAV team is then to complete these tasks in an effective fashion. This means that the team should be able to coordinate whilst interacting with such PDAs to *jointly* decide over an assignment of the tasks that can maximise the number of completed tasks whilst preserving the limited battery capacity of the vehicles. When all the tasks cannot be completed, it should also enable the UAVs to discriminate between the tasks so that the most important ones are preferred.

Now, coordinating UAVs whilst interacting with first-responders presents various challenges (see Section 3.2.1 for a more thorough description of the problem). Tasks can have various degree of importance. For instance, collecting imagery of a building on fire is typically more important than collecting imagery of a building that has just been evacuated. The termination of a task is uncertain. For instance, some tasks might require imagery to be collected for a long interval of time (e.g. a building on fire), whereas others might require it to be collected for a short interval (e.g. a burnt out building), but the exact time cannot be known with precision before-hand. UAVs have a limited battery and some of them can fail, but the remaining team should continue the operation nonetheless. Consequently, coordination should be performed in a *decentralised* fashion.

Against this background, Section 3.2 proposes a novel solution to the problem of coordinating teams of UAVs for dynamic real-time task assignment. We show how this problem can be cast into a decentralised optimisation problem and introduce a novel utility function in which several constraints, such as the task's importance and the UAVs' battery capacity, are carefully weighted to maximise performance (Sections 3.2.2.1 and 3.2.2.2). In so doing, we improve over canonical metrics such as entropy or mutual information gain which focus only on maximising the information collected since we incorporate *all* the above mentioned constraints into one single function. Next, we solve the coordination problem by using the max-sum algorithm and distribute the computation between both the UAVs and the PDAs so that the computation of the solution of the problem is not delegated only to the UAVs which are potentially the most unreliable part of the system (Sections 3.2.2.3, 3.2.2.4 and 3.2.2.5). Next, in Section 3.2.3, we empirically evaluate our utility against similar ones but which do not take all the constraints into

account. By so doing, we show that our function yields a better trade off between the quantity and quality of completed tasks. We then describe our experience of deploying our techniques on two real UAVs and show the effectiveness of max-sum in different scenarios, thus demonstrating its practical viability. Finally, Section 3.3 summarises the chapter.

3.1 The Max-Sum Methodology

Our methodology focuses on the incomplete version of the max-sum algorithm. Indeed, as discussed in Chapter 2, the incomplete max-sum algorithm requires less computation and communication than the complete versions of the algorithm. Thus it is more suitable to be deployed on low powered robotic agents such as unmanned aerial or ground vehicles (UAVs and UGVs respectively) which constitute the type of agents that we will deploy in this chapter. More specifically, as mentioned in Chapter 1, our methodology tackles problems related to the deployment of robotic agents in the domain of disaster response to acvhieve situational awareness.

Within this setting, we adopt the explicit coordination architecture discussed in Chapter 2. In more detail, the algorithm is not used as a single shot optimisation technique. Rather, it becomes a distributed message passing protocol embedded within the coordination architecture. This message passing protocol allows the agents to continuously make decisions as the mission proceeds, by calculating the utility that each of their possible joint decisions can yield. These utilities, as we will shortly see, are encoded within the function nodes of the factor graph. This means that the latter function nodes now represent the utilities of collective decisions, instead of constraints between variables as was discussed in Section 2.4.1.

In more detail, our methodology is composed of five steps. Each describes one aspect to take into account when deploying max-sum.

Step 1 – Defining Variables: Each variable of the factor graph represents a vehicle's decisions. These decisions take one of two forms in situational awareness domains:

- **Decisions as actions**: Each action is specific to the type of vehicle that is deployed. Each action is related to the manoeuvres that the vehicle can make and is defined by discretising its motion space. For instance, the actions of a fixed wing UAV are defined as the set of bank angles that it can follow, whereas the actions of a UGV are defined as its steering inputs.
- **Decisions as tasks**: Each task is a unit of work to be attended. Examples related to disaster management include imagery requests or tracking targets such as drifting life-rafts and ground vehicles.

The developer needs to be extremely careful when defining variables, since their number and the size of their domain influence the performance of max-sum in two ways. First, in terms of communication overhead, the length of a max-sum message to or from a variable x_i is linear in the size of its domain D_i ($O(|D_i|)$). Second, the complexity of computing a message from a function to a variable message is $O(\prod_{x_k \in} |D_k|)$. Thus, it is polynomial in the size of the individual domains, and exponential in the number of connected variables. The latter can be problematic when decisions represent tasks since their number is likely to be large. To address this shortcoming, a variant of max-sum exists, tailored for task assignment problems which was presented in Chapter 2).

In this work, the algorithm is used on a task assignment problem. Hence, this version is the one adopted here.

Step 2 – Defining Functions: Each function of the factor graph quantifies the impact of a joint set of decisions on the value of the global constraint function (Equation 2.1). These functions take one of two forms for situational awareness domains:

- **Utility of a vehicle**: This utility identifies the contribution of each vehicle to the set of measurements that the team is making. This representation is used in information gathering problems such as those described in Section 2.2. In particular, it is used when vehicles are coordinating to make collective measurements of some specific feature such as the temperature of a building or the position of a drifting life-raft (Bourgault et al., 2004). In these settings, actions represent a vehicle's decisions and are used to identify the local measurement corresponding to each of its possible manoeuvres. A collective measurement is then defined as the union of all the vehicles' local measurements.
- **Utility of a task**: This utility quantifies the assignment of one or more vehicles to a task (e.g. providing imagery of an area or tracking a drifting life-raft). A decision then represents the assignment of a task to a UAV. Thus, a utility assigns a value to determine which vehicle is more capable to attend a specific task, given the vehicle's properties such as its battery life or current position, and the task's properties such as its duration and importance.

Step 3 – Allocating Nodes: The computation related to the functions and the variables of the factor graph (algorithms 2.3 and 2.4) needs to be allocated to one of the available sources of computation. These include vehicles, laptops, desktops and PDAs. This is extremely useful since UAVs and UGVs have heterogeneous and limited computational resources and, most importantly, they can fail. Each node is then assigned as follows:

- **Variable Allocation**: Each variable is allocated to the vehicle whose decisions it represents. Another option is to allocate the variable to an independent platform such as a laptop or a PDA, considering that a very reliable communication channel needs to be built between the variable and the actual vehicle in order to send each decision.

- **Function Allocation**: Each function is allocated depending on which of the two approaches defined in step 2 is used:

  - **Utility of a vehicle**: Each function is allocated to the vehicle whose utility it represents.
  - **Utility of a task**: Each function can be allocated to any vehicle that can attend the task. Any allocation mechanism can be used to select one of these vehicles. For instance, a random allocation mechanism can be used or the one with the lowest id can be chosen.

  Note that, similarly to variables, each function can be allocated to an independent platform provided that a reliable communication channel is built.

Step 4 – Selecting a Message-Passing Schedule: A schedule for computing algorithms 2.3, 2.4 and to make a decision (Equation 2.11) is necessary for three reasons. First, max-sum requires the nodes to share a specific number of messages before Equation 2.11 can be calculated for each variable (i.e. before the vehicles can compute a decision). Second, in most real world settings, the structure of the problem changes continuously, therefore messages need to be shared to incorporate and propagate the changes in the environment. Third, communication is lossy in many real world settings, therefore sending redundant messages (i.e. computing algorithms 2.3 and 2.4 more frequently) can result in more messages being received. Three schedules can be used:

- **Synchronised schedule**: A node waits to receive all the messages from its neighbours before computing new ones or making a decision. This schedule can only be used in settings where communication is perfect (i.e. in simulation). When this is not the case, as in most real world settings, a node can wait for a message to arrive for an undefined amount of time. Thus it would generate a deadlock which would prevent all the other nodes from functioning.

- **Periodic schedule**: A node computes its messages (and makes a decision) periodically given the most recent messages. This schedule is highly recommended in scenarios where communication is lossy.

- **Response schedule**: A node sends a new message (and makes a decision) in response to the arrival of a single message from another node. This drastically reduces the computation of redundant messages compared to a periodic schedule.

Indeed, new messages are computed only in the presence of new information. However, since this redundancy is now lost, this schedule is suitable in domains where communication is not lossy.

Step 5 – Updating the Neighbourhood: The vehicles used in disaster management typically use broadcasting to communicate with each other, since it is cheaper and easier to implement than point to point or multicast communication (Grocholsky, 2002). However, recall from Section 2.4.4 that each node of the factor graph sends and receives messages from a specific set of neighbours. Due to the dynamism, these neighbours can change since the structure of the factor graph can vary continuously. Hence, each node needs to keep track of, and constantly update, its neighbours to identify its own messages among those that it receives. Depending then on the type of node, this update happens as follows:

- **Variable nodes**:
  - **Decision as actions**: For each vehicle, the neighbouring function nodes of its decision variable are the utilities of all the other vehicles that can make measurements that overlap with its own. In order to calculate these neighbours, all the measurements collected thus far are, initially, fused into a global estimate of a feature of interest, such as the temperature of a building or the radiations emanating from a power plant (Bourgault et al., 2004). Then, this estimate is used to calculate each vehicle's contribution to the next collective measurement (Stranders et al., 2010).
  - **Decisions as tasks**: For each vehicle, the neighbouring function nodes of its decision variable are the utilities of all the tasks that it can attend to. These are continuously updated by the communication between the different platforms, so that completed tasks are deleted while new ones are added.
- **Function nodes**:
  - **Utility of a vehicle**: For each vehicle, the neighbouring variable nodes of its utility are the decision variables of all the other vehicles that can make measurements that overlap with its own. These are calculated when updating the utility with new overlapping measurements.
  - **Utility of a task**: For each task, the neighbouring variable nodes of its utility are the decision variables of all the vehicles that can attend to it. These are continuously updated by the communication between the different platforms so that new vehicles that can attend the task are added, while those that can no longer attend are removed.

FIGURE 3.1: An example of a PDA used by the first responders within our setting.

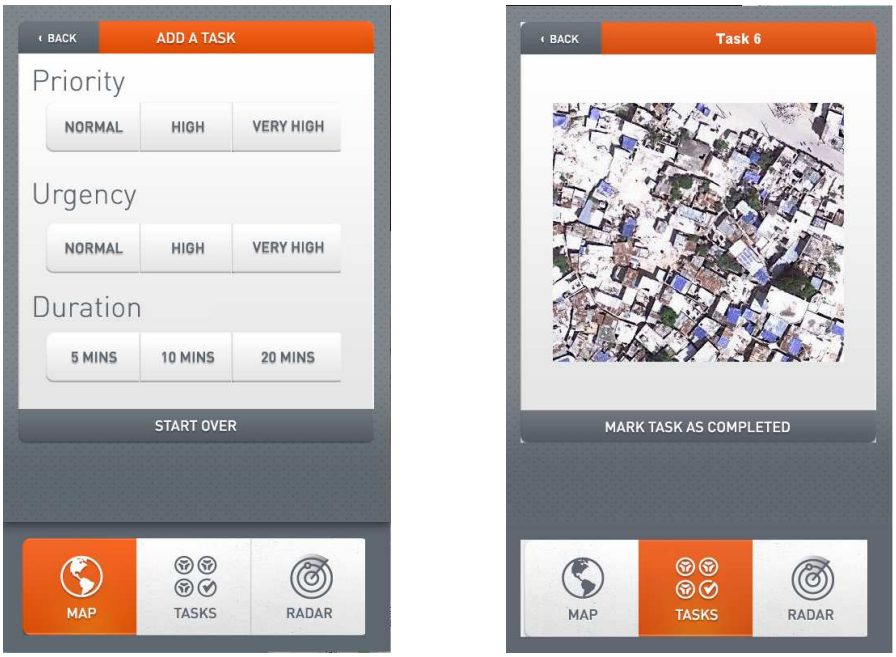
In order to understand the way in which this methodology can be used to deploy max-sum in real world settings, we will now outline a case study in which the algorithm is used to coordinate teams of UAVs for disaster management.

3.2 Case Study on Disaster Response

In this section, we describe our disaster response system that allows first responders to request imagery collection tasks to a team of UAVs flying above the area of a disaster.

3.2.1 Problem Description

The first responders request imagery collection tasks from the UAVs using a touch screen PDA (see Figure 3.1). This PDA runs a touch screen interface which allows a first responder to submit imagery tasks of any area within his proximity (see Figures 3.2(a) and 3.2(b)). Each task T_j represents a location (in geographic coordinates) for which imagery is required. To submit a task, each first responder sets three properties (Figure 3.2(a)): (i) priority $p_j = \{normal,\ high,\ very\ high\}$, representing the importance of the task (i.e. collecting imagery of an occupied building is more important than doing so for an empty one); (ii) urgency $u_j = \{normal,\ high,\ very\ high\}$ used to prevent tasks' starvation, as will be discussed shortly and (iii) duration d_j, which defines the interval of time for which imagery needs to be collected. Note that a first responder does not know this duration with precision since it depends on the specific reason for which imagery is required (e.g. to search for a casualty or to check access to an area). Thus, three estimates are considered ($d_j = \{5\text{ min},\ 10\text{ min},\ 20\text{ min}\}$). To complete a task, a UAV needs to fly to the specified location, station itself above it and stream live video to the PDA until the first responder indicates that the task is completed (Figure 3.2(b)). The



(a) Task Input (b) Task Monitor

FIGURE 3.2: The PDA's interface.

information about each submitted task is then broadcast by the corresponding PDA, so that the UAVs in the surrounding area that receive it add the task to the set of tasks that they can potentially attend. The UAVs then *jointly* decide which task each vehicle should complete and, in so doing, they maximise the number of completed high priority tasks. We achieve this coordination by applying max-sum as discussed next.

3.2.2 Application of the Methodology

The max-sum algorithm is used here to allow the UAVs to *jointly* decide which task each of them should attend. The algorithm is deployed as follows:

3.2.2.1 Step 1 – Defining Variables:

Each variable x_i takes values in the set of tasks that UAV i can attend. This choice follows from step 1 of the methodology, when decisions represent tasks. As a consequence, the domain of x_i is defined as the set of tasks $\mathcal{T}_i$ that UAV i can attend, where $\mathcal{T}_i \subseteq \mathcal{T}$ and $\mathcal{T} = \{T_1, T_2, \ldots, T_N\}$ is the set of all the submitted tasks. Since UAVs have limited communication capacities, they will only be able to attend a subset $\mathcal{T}_i$ of the tasks in $\mathcal{T}$. Again this follows from step 1 of Section 3.1.

3.2.2.2 Step 2 – Defining Functions:

A utility function U_j measures the utility of one or more UAVs attempting to complete task T_j. This choice follows from step 2 of the methodology applied to the case where utilities quantify the assignment of a task.

To achieve this, the key factor to model, in addition to the priority p_j and the urgency u_j, is a task's termination. As stated in Section 3.2.1, the termination depends on whether a first responder is satisfied with the amount of imagery collected. Hence, we can represent it as an uncertain factor of the utility function U_j. In particular, we model task completion as a Poisson process[1] (Chitale, 2008) measured over the time interval in which one or more UAVs can station itself above task T_j. To define this interval, consider *all* the UAVs that can attend T_j. Live imagery can then be collected from $t_1 = \min_i t_i^1$ to $t_2 = \max_i t_i^2$, where $t_i^1 = \frac{d_{ij}}{V_i}$ is the time[2] required by a UAV i to reach task T_j and where $t_i^2 = t + b_i$ is the remaining time that UAV i can remain on station (b_i is the UAV battery capacity, in terms of remaining flight time) above T_j. The utility U_j is then defined as follows:

$$U_j(X_j) = p_j \cdot u_j^{t-t_j^0} \cdot \left[1 - e^{-\lambda_j \cdot (t_2 - t_1)}\right] \tag{3.1}$$

where t is the current time, and p_j, u_j and t_j^0 are, respectively, T_j's priority, urgency and activation time.

Intuitively, the utility defined by Equation 3.1 measures the impact of each possible assignment $X_j' \in X_j$ of the UAVs aware of T_j on its completion (i.e. t_1 and t_2 are calculated over those UAVs where $x_i = T_j$). Each possible assignment then represents a different subset of the UAVs that can attend T_j and for which U_j needs to be calculated. In so doing, this utility allows the UAVs to make a variety of sophisticated decisions based on all the possible constraints of the problem. For instance, the platforms will always choose tasks with higher priority (due to the factor p_i in Equation 3.1). If these have the same priorities, the UAVs will always choose the one that has remained unattended for a longer interval of time (due to the factor $u_i^{t-t_i^0}$ in Equation 3.1)).

In addition, multiple UAVs may attend a task if this extends the time span for which at least one UAV is on station above the task. To understand this, consider the factor $1 - \exp^{-\lambda_j \cdot (t_2 - t_1)}$ of Equation 3.1. This factor represents the probability $P(t_1 \leq t_j^f \leq t_2)$ that the time at which T_j is going to terminate lies within the interval $[t_1, t_2]$. More

[1] $\lambda_j = \frac{1}{d_j}$ is the rate parameter of the Poisson process.

[2] d_{ij} is the Euclidean distance between a UAV i and task T_j and V_i is UAV i's average speed

formally, it is derived as follows:

$$P(t_1 \leq t_i^f \leq t_2) = \int_{t_1}^{t_2} \lambda_j \cdot \exp(-\lambda_j \cdot x) dx \tag{3.2}$$

$$= \int_{0}^{t_2 - t_1} \lambda_j \cdot \exp(-\lambda_j \cdot x) dx \tag{3.3}$$

$$= \lambda_j \cdot \left[\frac{-e^{-\lambda_j \cdot x}}{\lambda_j} \right]_0^{t_2 - t_1} \tag{3.4}$$

$$= 1 - \exp^{-\lambda_j \cdot (t_2 - t_1)} \tag{3.5}$$

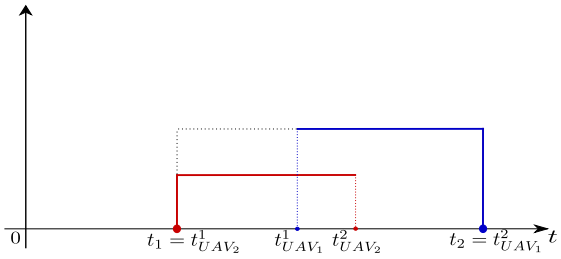
Hence, multiple UAVs will attend a task, when the time span ($t_2 - t_1$ in Equation 3.1) depends on more than one UAV. For instance, whenever, one UAV has a higher remaining battery life (i.e. t_2 is higher) but another one is closer to the task (i.e. t_1 is smaller)

In general, the key advantage of this utility is that it allows the UAVs to coordinate over a variety of sophisticated decisions which take into account the multiple parameters of the problem, such as the UAVs battery capacity, their distance to the tasks and the task's actual importance. The following example illustrates one such behaviour:

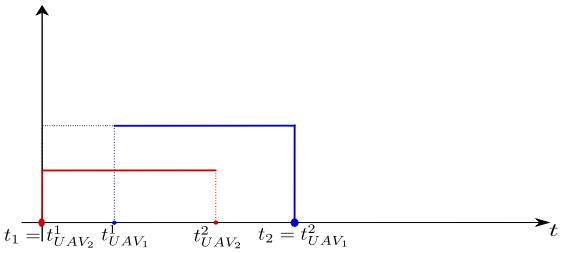
Example 3.1. ***Task Handover****: Consider a scenario with two vehicles UAV_1 and UAV_2. UAV_1 has a long remaining battery capacity ($b_1 = 1000s$), while UAV_2 has less ($b_2 = 800s$). Two tasks T_3 and T_4 are submitted to the system: T_3 has a high priority ($p_3 = high$) and T_4 has a low one ($p_4 = low$). Both the tasks have low urgency and duration. The time required by the UAVs to reach each of the tasks is shown in the following table:*

	UAV_1	UAV_2
T_3	$80s$	$70s$
T_4	$60s$	$50s$

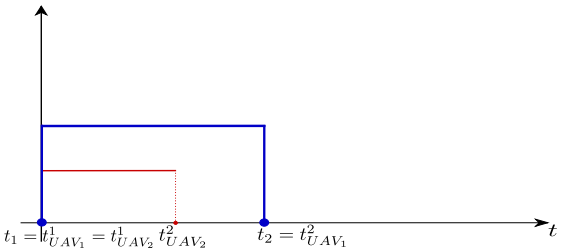
Figure 3.4 illustrates the scenario. Given this setting, the utility (Equation 3.1) for both T_3 and T_4 will be higher when both the UAVs attend each task ($t_1 = 70s$ for T_3 and $t_1 = 50s$ for T_4 whereas $t_2 = 100s$ for both T_3 and T_4). To understand this, consider the time interval shown in Figure 3.3(a) for task T_3 (the same interval applies to T_4): the interval of time $[t_1, t_2]$ on which the probability of the tasks depends (see Equation 3.1) is higher when t_1 belongs to UAV_2 and t_2 belongs to UAV_1. This means that Equation 3.1 is maximised when both the UAVs attend the task. Task T_3 will then be selected first since it has a higher priority. Next, as shown in Figure 3.3(b), UAV_2 will reach T_3 first since it is closer, but the allocation of UAVs to tasks will remain the same. However, this changes as soon as UAV_1 reaches T_3 (Figure 3.3(c)). Indeed, as shown in the figure, the interval $[t_1, t_2]$ is now larger when both t_1 and t_2 belong to UAV_1. This means that



(a) UAV_2 is closer to T_3 ($t_1 = t^1_{UAV_2}$ in Equation 3.1) but UAV_1 has a higher battery life ($t_2 = t^2_{UAV_1}$ in Equation 3.1)



(b) UAV_2 reaches T_3 before UAV_1, the allocation remains the same, due to Equation 3.1.



(c) UAV_1 also reaches T_3: the allocation changes and the task is handed over to UAV_1 since $t_1 = t^1_{UAV_1}$ and $t_2 = t^2_{UAV_1}$ in Equation 3.1

FIGURE 3.3: Three sequential configurations of the remaining battery capacity (t_2) and the time to reach the task (t_1) (see Equation 3.1) of the UAVs for task T_3 in Example 3.1.

the utility of T_3 is now maximised only when UAV_1 attend the task. Hence UAV_2 will hand over the task to UAV_1 and will head over to task T_4.

3.2.2.3 Step 3 – Allocating Nodes:

Each UAV is allocated the variable representing its decisions. Similarly, each PDA is allocated the utility functions of the tasks submitted by the first responder.

The reasons for doing this were discussed in step 3 of Section 3.1. In essence, by choosing such approach, we use all the available computational power (of both UAVs and PDAs) and, additionally, we reduce the UAVs responsibilities in controlling the team's decision making process which guarantees more robustness. Indeed, as discussed in Chapter 2,

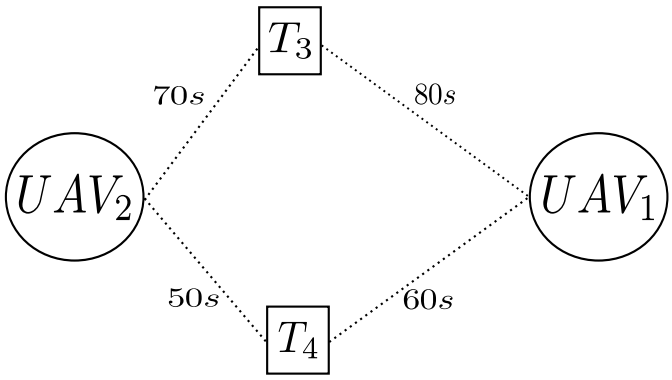


FIGURE 3.4: The scenario of two UAVs and two tasks illustrated in Figure 3.1

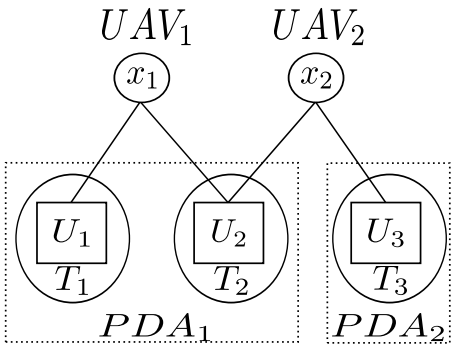


FIGURE 3.5: A factor graph showing 2 variables nodes, 3 function nodes and the platforms controlling them.

the UAVs are, potentially, the most unreliable part of our system. Thus delegating all the computation to them might not be the safest choice. Figure 3.5 shows an example of a factor graph resulting from this allocation. The figure shows two UAVs (UAV_1 and UAV_2) controlling two variables x_1 and x_2 and two PDAs: PDA_1 controls two tasks T_1 and T_2 (and the corresponding utilities U_1 and U_2) while PDA_2 controls one task T_3 (and its utility U_3).

3.2.2.4 Step 4 – Selecting a Message-Passing Schedule:

We use a periodic schedule to compute the max-sum messages. This choice follows from step 4 of Section 3.1. Each device (UAV or PDA) then periodically runs algorithms 2.3 and 2.4 to compute the new messages and decisions (Equation 2.11), given the messages that it received.

3.2.2.5 Step 5 – Updating the Neighbourhood:

Within our setting, variables can appear and disappear at anytime since UAVs can run out of battery and new tasks are constantly submitted or completed. Thus, as suggested

by step 5 in Section 3.1, UAVs and PDAs continuously share information about their status (e.g. their positions, the UAVs' remaining battery and the tasks' properties). Then, the information about each variable's neighbours is stored and updated as new tasks are submitted or completed. Similarly, the information about each function's neighbours is updated each time a new UAV becomes able to complete the function's task or runs out of battery life.

The use of our methodology allows max-sum to handle both the coordination between the different UAVs and their interactions with the PDAs at the same time. In so doing, we avoid the use of distributed protocols, which would handle the coordination and the interaction separately. In terms of decision making, this means that UAVs continuously produce new collective decisions by calculating the joint variable assignment X^* that maximises the sum of all the tasks' utilities:

$$X^* = \arg\max_X \sum_{j=1}^{N} U_j(X_j) \tag{3.6}$$

The previous equation is a specialisation of Equation 2.1 to the task assignment setting.

After completing step 5, the system is now ready to be deployed. In the next section we present our empirical evaluation.

3.2.3 Empirical Evaluation

The aims of our experiments are twofold. First, we wish to study the properties of the task utility and verify whether it is actually effective for solving the task allocation problem. In more detail, we wish to verify whether the utility allows the UAVs to carefully trade-off between the different tasks and always try to complete the tasks with the highest priority. To achieve this, we benchmark our utility against similar ones in a controlled repeatable setting (Section 3.2.3.1). The latter utilities behave greedily and abstract out some of the constraints of the problem to favour a quicker allocation of the tasks.

Second, we deploy our system on two "hexacopters" (see Chapter 2 for more details) and evaluate its performance in the real world (Section 3.2.3.2). In so doing, not only are we able to ascertain how our coordination system responds to the complexity and the unpredictability of the real world, but also, we can verify the actual performance of max-sum as a coordination algorithm for robotic agents.

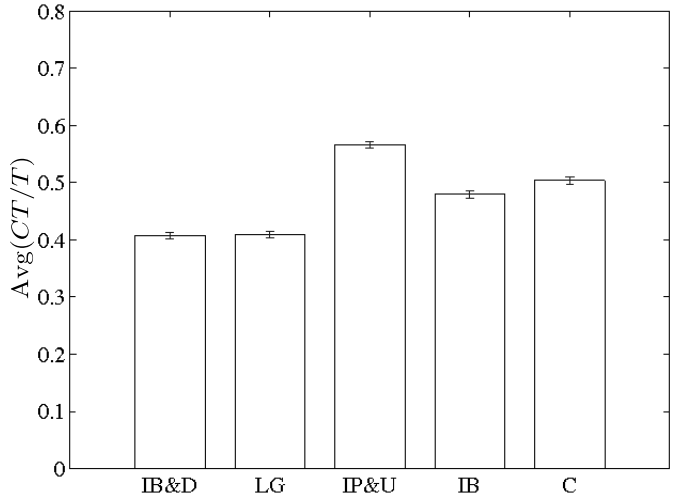
Within our experiments, we make two asssumptions. First, we assume that neither the UAVs nor the PDAs can fail. The reason for this is that we reproduce this assumption within our tests because variables and functions continuously appear and disappear due

to either the completion of a task or one UAV running out of battery life. Second, we assume that communication between the UAVs and the PDAs is perfect (i.e. no messages are lost and the delay between sending and receiving a message is limited to a few milliseconds). In fact, we decided to focus on testing the system assuming perfect communication for two key reasons. The first comes from a feature of max-sum discussed in Chapter 2. Intuitively, each max-sum message carries more information than other coordination algorithms. Hence, it is sufficient for very few messages to be received to yield good solutions. In addition, as discussed in Section 3.1, when communication is limited, a response schedule can be used to send a larger number of redundant messages to increase the probability of the agents receiving them. The second reason comes from literature which has shown that max-sum is robust to communication failure in simulated environments (Farinelli et al., 2008). Naturally, this still remains to be demonstrated in the real world and will be one of the challenges to address in our future work (see Chapter 8).

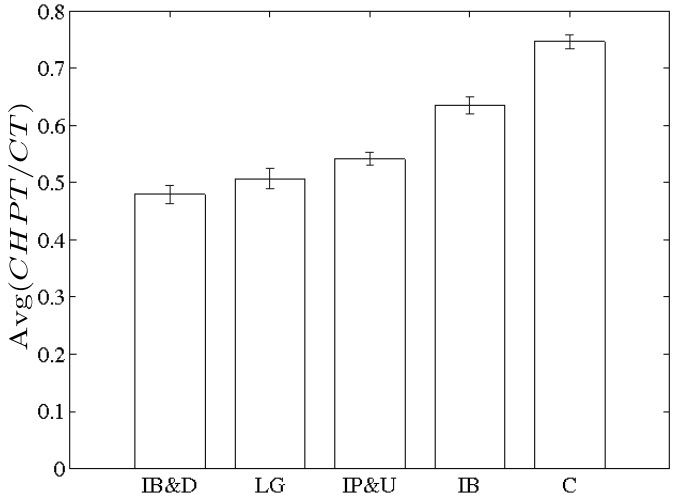
3.2.3.1 Simulation Tests

We benchmark our utility (the Complete utility C) against four other utilities, each considering an increasing number of constraints: (i) $U_i^1 = p_i \cdot u_i^{t-t_0^i}$ (ignores battery and duration $IB\&D$) that removes the constraint related to the duration (λ_i) and to the UAVs properties (i.e. remaining battery capacity t_2 and distance t_1); (ii) $U_i^2 = 1 - e^{t_2 - t_1}$ (ignore priority and urgency $IP\&U$) that removes the priority p_i, the urgency and the activation time constraint $u_i^{t-t_0^i}$; (iv) $U_i^3 = 1 - e^{t_{\max} - t_1}$ (ignore battery IB) that again removes priority and duration, but also the battery capacity ($t_{\max}$ is equal to the initial amount of each UAV's battery capacity); finally, as a lower bound we benchmark all the former approaches against a local greedy algorithm (LG) where each UAV selects which task to attend depending on the utility and (v) $U_i^5 = p_i \cdot u_i \cdot \frac{t_{\max} - t_{ij}}{t_{\max}}$ that gives a higher value to the closest high priority tasks.

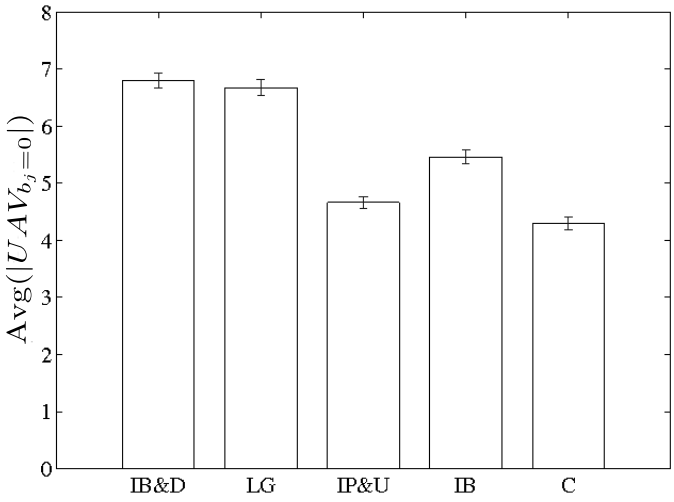
We run 200 simulations, considering a set of fixed independent variables whose values are chosen so as to generate settings likely to happen in the real world (Cole, 2009; Grocholsky, 2002). In more detail, we consider teams of 10 UAVs and 30 tasks, their initial position was randomly chosen within an area of 1500×1000m. For each UAV j: $b_j \in [500s, 1000s]$), $V_j = 5\ m/s$ and j's communication range is set to $400m$. For each task T_i: $t_i^0 \in [0, 1000]$); $p_i = 1$ or $p_i = 1000$; $d_i \approx 5min$ or $d_i \approx 20min$ (note that we only consider two values to emphasize the impact of the priority and the duration on each task's utility); finally, since both priority and urgency linearly influence the utility, we set the urgency ($u_i = 1.001$). The completion of a task is uncertain and drawn from a Poisson distribution. In each experiment we measure the number of completed tasks over the total number of tasks and the number of completed high priority tasks over the number of completed tasks. Finally, we measure the influence of each utility on the



(a) Average number of completed tasks over total submitted tasks (Avg(CT/T))



(b) Average number of completed high priority tasks over total completed tasks (Avg($CHPT/CT$))



(c) Average number of UAVs that run out battery while hovering above a task (Avg($|UAV_{b_j=0}|$))

FIGURE 3.6: Experimental Results

battery capacity of each UAV by recording the average number of time that each UAV runs out of battery capacity while hovering above a task.

Results are shown in Figures 3.6(a), 3.6(b) and 3.6(c). The error bars in the figures represent the standard error of the mean. In more detail, Figures 3.6(a) and 3.6(b) confirm that our utility does yield indeed a better trade off between the quality of tasks completed and their quantity. To understand this note that Figure 3.6(a) shows that the utility that yields the highest average number of completed tasks ratio is $IP\&U$ (55% of the tasks are completed). This is expected since it represents the less constrained utility, that does not discriminate between tasks and thus allow a higher number of them to be completed. Our utility and IB are more constrained, therefore they complete respectively 50% and 45% of the available tasks. The two remaining utilities ($IB\&D$ and LG) behave greedily and therefore by using the UAVs end up attending always the same tasks. Hence, they perform much worse than the other ones.

Now, Figure 3.6(b) shows that our utility is the one completing on average the highest number of high priority tasks (75% of the completed tasks have a high priority). Indeed, by using $IP\&U$ only 55% of the completed tasks have a high priority. Thus, through the use of our utility, the UAVs yield 5% less completed tasks than through the use of $IP\&U$. Thus these results show that our utility yields a better trade off between the quality and the quantity of the tasks completed. The other benchmarks perform as expected, by using IB, the second most constrained, 65% of the completed tasks have a high priority, whereas the remaining two ($IB\&D$ and LG) utilities yield roughly 50% of high priority completed tasks. This surprisingly high rate is explained by considering that the assignment of the priorities is based on a uniform probability (i.e. 50% of the tasks have a high priority and 50% have a low one).

Figure 3.6(c) shows that by using our utility, only 4 out of 10 UAVs run out of battery while hovering above a task, whereas by using the other utilities the number varies from 5 to 7. Running out of battery while hovering is a clear symptom of the fact that the assignment has been done in a non-effective fashion since the battery life of the UAV has not been taken into account. In this sense, the poor performance of the utilities not considering the battery life, such as LG, $IB\&D$ and IB is to be expected, whereas the performance of the two remaining utilities is roughly the same since they both take the UAVs battery into account.

3.2.3.2 Real-World Flight Tests

We demonstrate our system using two commercial off-the-shelf Mikrokopter hexacopter multi-rotor rotary wing UAVs (see Chapter 2) and two ground-based PDAs. These were deployed on two desktop PCs (Intel Core 2 Duo 3.0GHz, 3.2Gb RAM) connected

via ethernet. Figure 3.7 illustrates the way we built our system. As shown in the figure, a software module containing the max-sum algorithm described in Chapter 2 is implemented for both UAVs and PDAs. Each decision made by a UAV is transmitted to the corresponding hexacopter wirelessly. Each UAV is provided with a flight control system that provides both attitude stabilisation, as well as GPS waypoint-based guidance system to control its motion—it follows a sequence of waypoints representing locations to reach—while holding a pre-determined altitude. A pair of $900MHz$ radio modems are used to establish a wireless command and control datalink between the ground control software and the UAVs. Each UAV is finally equipped with a downward pointing video camera to capture imagery of the targets (white squares laid on the ground).

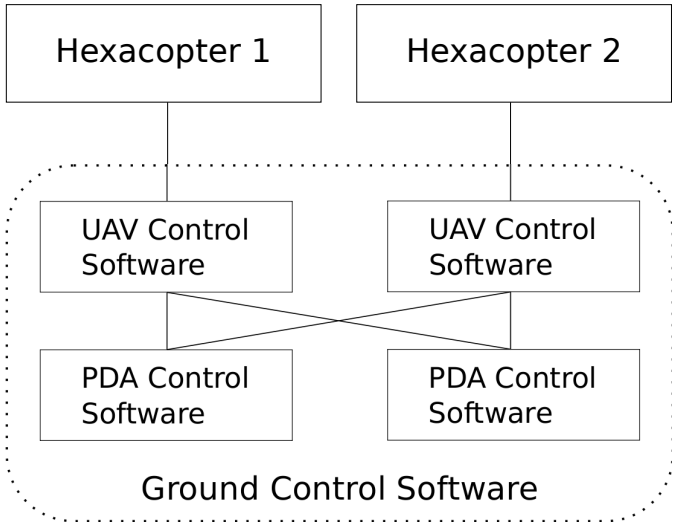


FIGURE 3.7: The architecture of our System

Our tests were run at a test facility outside of Sydney, Australia. A video summarising the tests can be found at `http://vimeo.com/34800379`. In the video (see Figure 3.8 for a snapshot), windows A and B show the hexacopters, window C shows the computation over the factor graph over which max-sum is running and window D shows the path of the UAVs. We conducted three tests:

Flight 1 – Homogeneous Tasks: Two identical tasks (T_1 and T_2 in Figure 3.9(a), both have normal priority and urgency, 5 min duration) are simultaneously submitted to the UAVs (UAV_1 and UAV_2 in the figure). The aim of this test is to assess the behaviour of the coordination mechanism in response to a canonical coordination scenario. In this setting, the maximum of each task's utility is obtained when the task is assigned to the closest UAV (this is due to the exponential factor in Equation 3.1). Initially, the two UAVs coordinate by sharing max-sum messages with the PDAs. The coordinated decision that maximises the sum of the tasks' utilities $U(\mathbf{x})$ is then the one in which each UAV is assigned a single task. Indeed, this is what we observed during our test

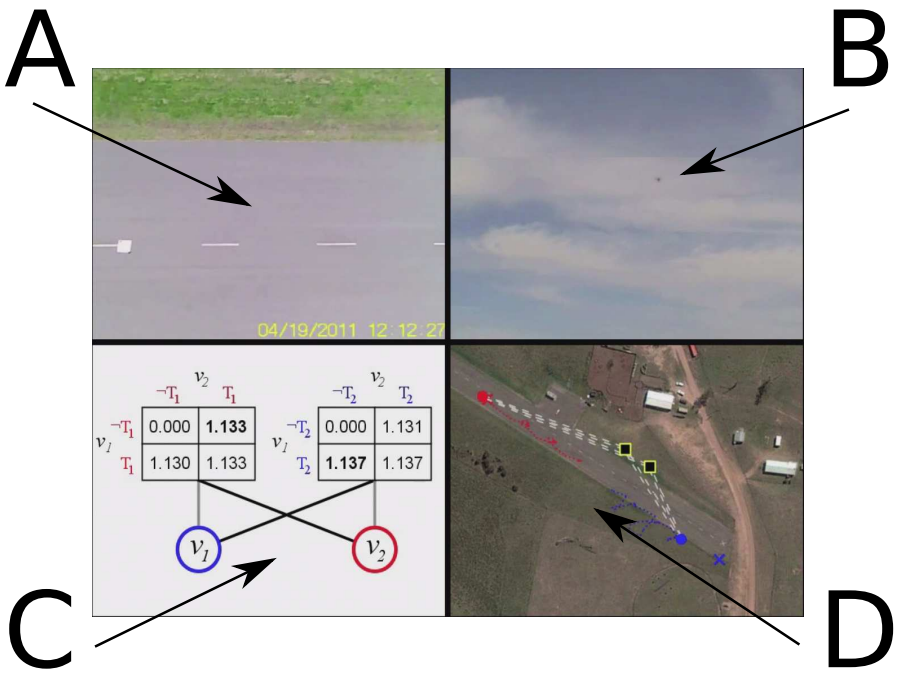
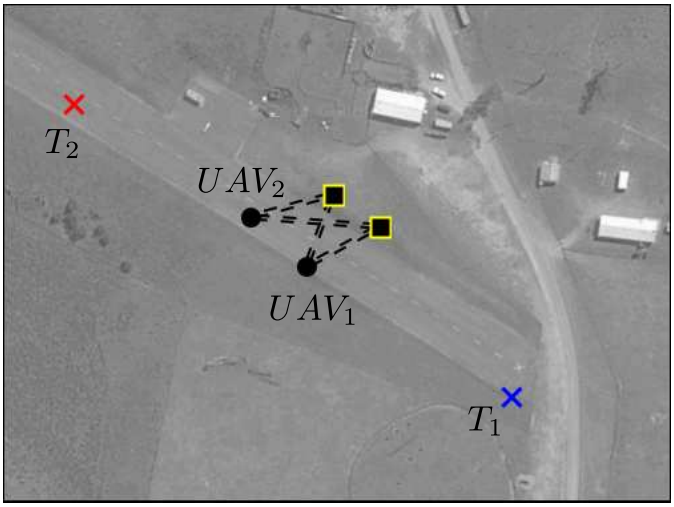


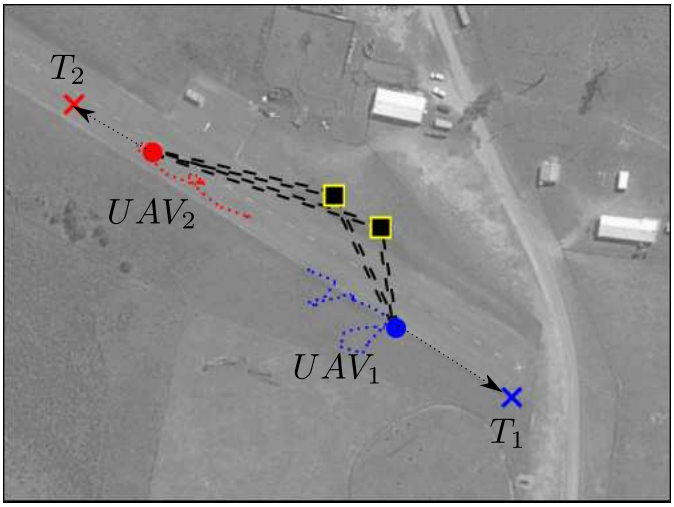
FIGURE 3.8: A snapshot of the video summarising the three flight tests.

3.9(b), confirming the correctness of our system. Figure 3.9(c) finally shows the UAVs hovering above their corresponding tasks.

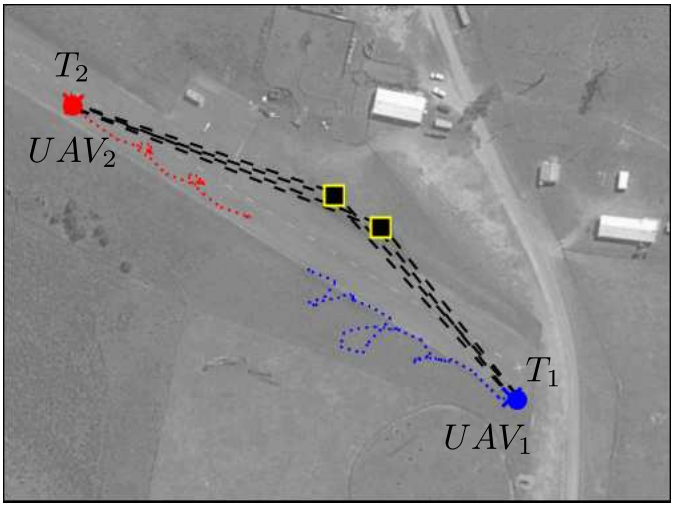
Flight 2 – Sequential arrival of Tasks: Two different tasks (T_1 and T_2 in the figures, T_1 has a normal priority, while T_2 has a high priority, both have normal urgency and 5 min duration) are submitted to the UAVs (UAV_1 and UAV_2 in the figures). T_2 is submitted 40s after T_1. The aim of this test is to assess the behaviour of the mechanism in the presence of heterogeneous properties and dynamism. Initially, only T_1 is present and the maximum of its utility is obtained when it is assigned to both the UAVs (due to the exponential factor in Equation 3.1). Note that the utility of assigning the task to both the UAVs cannot be lower than the utility of assigning it to only one of them (i.e. as shown by Equation 3.1). For this reason, in this case, the UAVs both go to the same task (Figure 3.10(a)). As soon as T_2 appears, the setting becomes the same as per flight 1. Thus, the maximum of each task's utility is obtained when the task is assigned to the closest UAV. Thus, the UAVs revise their decisions and UAV_2 goes to complete it (Figure 3.10(b)). In so doing, this setting also demonstrates the advantage of a coordination approach such as max-sum which allows the UAVs to complete both the tasks, instead of a greedy algorithm which would not allow the agents to revise their decisions and would keep sending them to the same task, resulting in a sub-optimal allocation. Next, once T_1 is completed, the setting becomes the same as the beginning of Flight 2. Thus, again the UAVs revise their decisions and are both assigned to the remaining task (Figure 3.10(c)). Three coordinated decisions then maximise the sum of the tasks utilities $U(\mathbf{x})$. Initially, the best decision is the one in which both the UAVs



(a) Flight 1: The UAVs make a decision

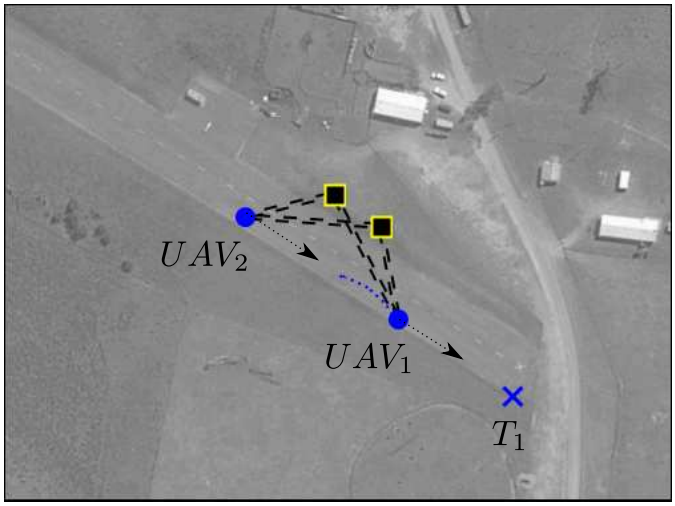


(b) Flight 1: The UAVs move to their tasks

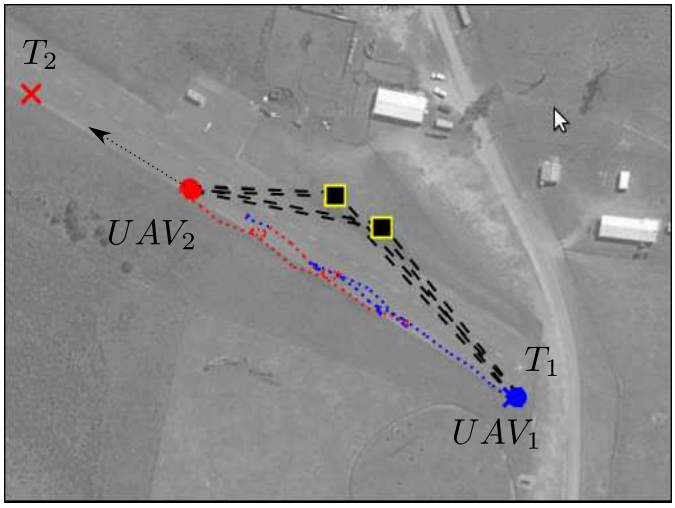


(c) Flight 1: The UAVs complete their tasks

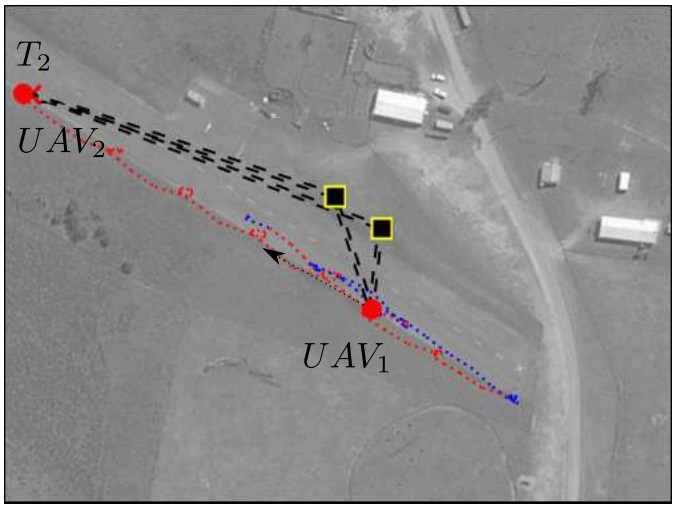
FIGURE 3.9: A sequence of snapshots depicting the behaviour of the two UAVs in flight 1.



(a) Flight 2: The UAVs go to the first task



(b) Flight 2: The second task appear, the UAVs change their decisions



(c) Flight 2: First task is completed, the UAV goes to the second task

FIGURE 3.10: A sequence of snapshots depicting the behaviour of the two UAVs in flight 2.

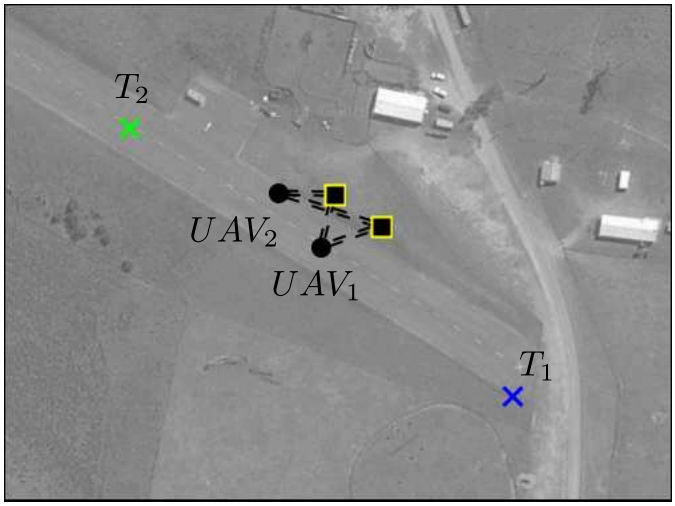
are assigned to the only available task. Then, the best decision becomes the one in which each UAV is assigned a single task. Again, this is what we observed during our test. Finally, the best decision is the one in which they are again assigned to the remaining task.

Flight 3– Heterogeneous Capabilities of the UAVs: Two identical tasks (T_1 and T_2, both with normal priority and urgency, 5 min duration) are submitted to the UAVs. However, here, UAV_2 receives the information only about T_2, while UAV_1 receives the information about both. After 60s a new task (T_3 with the same properties as the previous ones) is submitted to both the UAVs. The aim of this is to test the behaviour of the system when the capabilities of the UAVs are heterogeneous. Initially, only one assignment is possible since UAV_2 can only attend T_2. Thus, the maximum of this task's utility is obtained when the former UAV is assigned to it (Equation 3.1). The same applies for UAV_1 and T_1. Figure 3.11(a) shows such a situation in which the UAVs coordinate and go to one task each. As soon as T_3 appears, as per flight 1, the maximum of its utility is obtained when it is assigned to the closest UAV (UAV_2), which is, however, already completing another task (Figure 3.11(b)). Thus, two coordinated decisions maximise the sum of the tasks utilities $U(\mathbf{x})$. Initially, the best decision is the one that assigns each UAV to a single task. However, as soon as one UAV completes its task, the best decision becomes the one in which this UAV is assigned to the new task (Figure 3.11(c)).

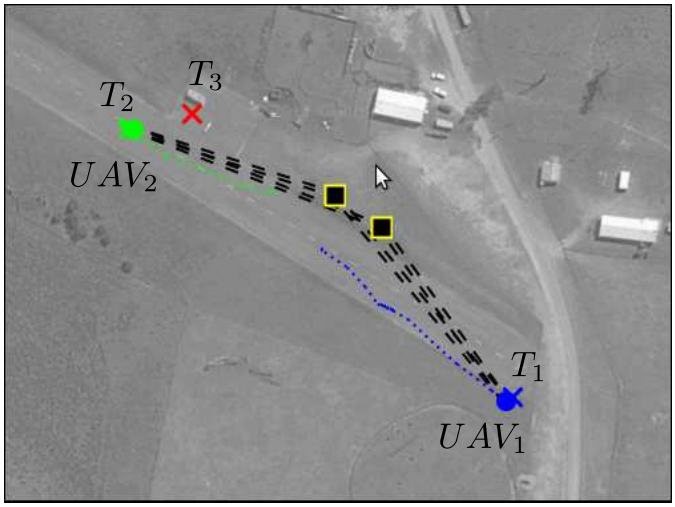
3.3 Summary

This chapter presented a study of the deployment of the max-sum algorithm to achieve situational awareness in disaster response scenarios. In more detail, we described a potential system for disaster response. The system uses the max-sum algorithm to allow a team of UAVs to interact with the first responders at the scene of a disaster in real time, and coordinate to complete imagery collection tasks. A novel task utility is defined so that the various constraints characterising the problem, such as the task's importance and the UAVs' battery capacity, are carefully weighted to optimise performance. By running max-sum, the system is decentralised and meets most of the requirements presented in Chapter 1. In particular, the system can scale because the computation is distributed between UAVs and PDAs, it is robust to component failure because there exists no central point of control and it is resource aware since the max-sum algorithm exploits the locality and the sparseness of the agents interactions to reduce the computation required to compute and send messages.

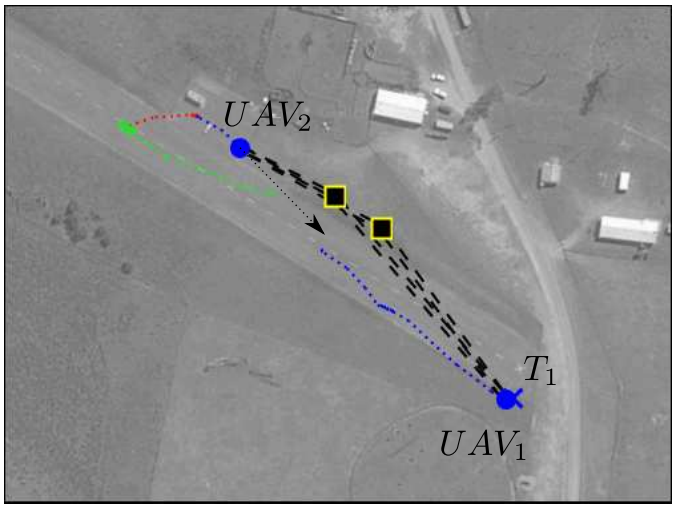
At the beginning of this chapter, we also argued that a user who is not familiar with max-sum and the generalised distributive law framework, might have trouble deploying the



(a) Flight 3: Two tasks only, the UAVs make a decision



(b) Flight 3: The third task appears



(c) Flight 3: Two tasks are completed, the UAV heads to task remaining

FIGURE 3.11: A sequence of snapshots depicting the behaviour of the two UAVs in flight 3.

algorithm on real robotic agents such as UAVs and UGVs. To address this shortcoming, this chapter proposed a methodology that carefully guides the less-experienced developer to a successful deployment of the max-sum algorithm for problems related to situational awareness. This methodology is a set of general rules that unifies the different ways in which the algorithm can be applied to these problems. In so doing, the algorithm can meet the generality requirement described in Chapter 1, because we define a set of rules to deploy it successfully for any problem related to situational awareness (see Chapter 2).

The system is evaluated both in simulation and in the real world. The former demonstrates that the use of our utility allows the UAVs to carefully trade-off between the different tasks and always favour the high priority ones given the remaining constraints of the problem. In so doing, it allows us to confirm empirically the accuracy of our system. As a consequence, it allows us to demonstrate the system's full effectiveness, despite the fact that it does not meet the performance guarantee requirement (Chapter 1). Real world tests are then used to ascertain the performance of the system in the real world. In more detail, the system is deployed on two hexacopters UAVs and is tested on three different settings. Our tests indicate that our system responds well to the complexity and the unpredictability of the real world. In addition, they show max-sum's power as an algorithm to coordinate real UAVs. Hence, since we argued in Chapter 1 that max-sum's potential for coordinating teams of real robotic agents still remains to be verified, by running these tests we can finally demonstrate max-sum's practical viability.

Despite this potential, however, max-sum still does not fully address the requirement of the complex interactions presented in Chapter 1. The focus of the next chapters is then on extending the algorithm to meet this requirement and on presenting three specific examples of complex interactions, namely multiple objectives (Chapter 5), uncertainty (Chapter 6) and online learning (Chapter 7). To do so, the first step, is to go back to pure theory and derive new formalisms and algorithms to model and solve coordination problems involving complex agents' interactions. This is the focus of the next chapter.

Chapter 4

A Class of Algorithms for Partially Ordered Coordination Problems

This chapter presents a general framework to represent settings in which agents are required to make complex joint decisions which cannot be measured using the real-valued constraint functions used in traditional distributed constraint optimisation problems (DCOPs). In so doing, this chapter provides the first fundamental step towards addressing the requirement of complex interactions described in Chapter 1.

In general, DCOPs constraint functions are typically used to measure the agents' joint decisions. Examples include the utility of a task assignment or the impact of a set of agents' actions on a collective measurement (see Chapter 3 for more details). However, in many application domains, the agents are typically interacting over complex decisions defined over multiple, different, dimensions. To measure the quality of these complex interactions, it becomes then necessary to trade off between such dimensions. As an example, consider the domain of disaster response discussed in Chapter 2. When a team of unmanned aerial vehicles (UAVs) is deployed to search and track lost life-rafts drifting at sea, they are making decisions over two parameters representing the former two objectives. Hence, they should carefully weight each of those objectives to make an effective decision (see Chapter 5). Similarly, uncertainty characterises most of the UAVs' parameters. For instance, the agents' location is not known with precision and their sensors are noisy. When this uncertainty is known, the UAVs can make decisions by trading off a set of parameters such as the mean, the variance and the risk characterising the former uncertainty (see Chapter 6). In contrast, when this uncertainty is unknown and the agents have to learn online the outcome of their decisions, considering parameters

such as the average sample value of the former decisions' outcome and their bounds (see Chapter 7).

Unfortunately, however, real-valued functions cannot encompass the complexity of the above mentioned parameters. For instance, a real valued function cannot model the trade off between searching and tracking targets for different UAVs because these two objectives are in fact conflicting. This means that maximising one objective might result in detrimental performance in terms of the other and this cannot be measured by a real valued function. In addition, it cannot represent the trade off between the uncertainty characterising the decisions of a team of agents, because it depends on the probability density function characterising such decisions and their risk. However, to define the risk it is necessary to carefully weight the uncertainty over the agents decisions, by considering all the possible parameters that define the specific pdf of the agents' decisions (i.e. it is necessary to consider a sufficient statistic). This, again, cannot be represented using a real valued function. Finally, a real valued function cannot model the trade off between the average sample value and the bounds on the uncertainty of the agents' decisions when they are learning online. The reason is that to define these bounds it is necessary to define the uncertainty related to the sampled value by using a set of parameters which are not measurable by simply using a real number.

Given these shortcomings, it becomes necessary to define a new framework capable of encapsulating of the different features defining coordination problems in which the agents are making complex interactions involving trading off multiple parameters. In addition, it becomes necessary to define new solution techniques capable of solving these problems, whilst trying to meet as many of the requirements defined in Chapter 1 as possible.

Now, trading off the parameters of the agents' interactions consists essentially in defining a way to compare several problem solutions and understanding which ones are better. In the case of real values, as for constraint functions in traditional DCOPs, this is fairly simple since these values are totally ordered and, for this reason, there always exists an optimum (a maximum or a minimum). However, when considering complex interactions this total order might not exist anymore. To understand this, consider the set of bi-objective couples $(2, 1)$, $(1, 2)$ and $(1, 1)$. Referring back to the search and track example defined above, these numbers could represent the values for a UAV for searching and for tracking different targets. Intuitively, we can say that both $(2, 1)$ and $(1, 2)$ are "better" than $(1, 1)$ (we will use the term "dominate" to represent the concept of "better than", as we will discuss in more detail in the following sections), since in both cases at least one value is better than the other. However, nothing can be said about $(1, 2)$ and $(2, 1)$. In essence, the former vectors are non-comparable and need to be considered as equivalent. In general, such vector functions are said to be *partially ordered* (Rollón, 2008). Thus, given a set of vectors, a partial order means that a single maximum does not necessarily

exist anymore. Rather, there is a set of multiple *non-dominated* alternatives which cannot be compared one with the other and which all need to be considered as potential solutions.

Against this background, the contributions of this chapter are twofold. First, we propose a novel framework to model coordination problems in which the agents are making complex decisions whose outcomes are measured using partially ordered functions. In more detail, in Section 4.1, we describe the framework of partially ordered distributed constraint optimisation problems (PO-DCOPs). A PO-DCOP generalises a DCOP by defining the constraint functions as partially, rather than totally ordered. In so doing, complex interactions involving trading off between heterogeneous parameters can be represented. As a consequence, our framework allows us to represent the coordination problems presented in Chapter 2, in a much more realistic way than canonical DCOP. Second, we propose the partially ordered generalised distributive law (PO-GDL) a semi-ring which can be used to instantiate coordination algorithms to solve PO-DCOPs based on the GDL, by using a procedure similar to the one described in Chapter 2. More specifically, the GDL exploits the algebraic structure of the global constraint function of an optimisation problem (i.e. a commutative semi-ring) to instantiate a message passing procedure, composed of two algorithms (Algorithms 2.2 and 2.1), which can be used to solve DCOPs. In so doing, we exploit the well known flexibility of the GDL to derive efficient optimisation algorithms, such as DPOP, Action GDL and bounded max-sum to design algorithms to solve PO-DCOPs. Hence, we propose a novel class of decentralised algorithms which can be used to solve coordination problems involving complex interactions and decisions. Three instantiations of the PO-GDL will then be used in the following chapters, to solve DCOPs involving complex interactions such as multiple objectives, uncertainty and online learning.

The remainder of this chapter proceeds as follows. Section 4.1 describes the general PO-DCOP framework. Section 4.2 then illustrates the methodology we defined to derive algorithms that solve PO-DCOPs by building upon the GDL. Section 4.3 illustrates some key properties of these algorithms including their optimality and their complexity. Finally, Section 4.4 summarises the chapter.

4.1 Partially Ordered Constraint Optimisation Problems

A PO-DCOP is a framework that generalises canonical DCOPs to settings in which the agents' joint decisions are measured by vector functions. More specifically, the constraints C_j in Definition 2.1 are generalised to vector functions. Formally, a PO-DCOP is defined as follows:

Definition 4.1. A PO-DCOP is a tuple $\langle A, X, D, F, C \rangle$ such that:

- $A = \{1, \ldots, |A|\}$ is a set of agents.
- $X = \{x_1, x_2, \ldots, x_n\}$ is a set of variables.
- $D = \{D_1, \ldots, D_n\}$ is a set of discrete domains, where D_i is the domain of variable x_i.
- $F : X \to A$ is a function that assigns variables to agents. Each agent controls (the value of) the variables that are assigned to it.
- $C = \{C_1, \ldots, C_m\}$ is a set of partially ordered constraint functions. Each constraint C_j is a vector function mapping assignments of X_j to $\mathbb{R}^k$ (k is the arity of the functions in C).

Thus, in a PO-DCOP the sets A, X, D, and F maintain the same meaning as in traditional DCOPs. In contrast, the constraint functions in C are generalised into partially ordered constraint functions. In practice, they are defined as vector functions, where each vector contains the different parameters that are necessary to formalise an interaction between different agents. For instance, we will see in Chapter 5 how these vectors are defined over multiple objectives. In Chapter 6 then, we will see how the vectors are defined over sufficient statistics that represent the probability density function characterising the uncertainty over the agents' decisions (e.g. mean and variance for Gaussian distributions). Similarly, in Chapter 7, we will see how the vectors will be defined over the sample value and the bound defining again the stochasticity over the agents' decisions. In light of this, we can state the following property:

Property. A DCOP is a PO-DCOP where each constraint function is totally ordered and where each vector function $C_j \in C$ is defined over one single parameter.

To understand this property, note that a total order defined over a set of generic instances S is a specific type of partial order such that each element in S can actually be compared (i.e. ordered). In addition, considering that a PO-DCOP is still an optimisation problem, its solutions are the ones that optimise (maximise or minimise) the problem. In light of this and of the previous property, the solutions of a PO-DCOP are the ones that optimise the aggregation of the constraint functions, given an optimisation criterion:

$$X^* = \arg \max_{X}^{\succ} \bigotimes_{j=1}^{m} C_j(X_j) \tag{4.1}$$

In Equation 4.1, the $\otimes$ operator is used to aggregate the different vector functions. It is a generalisation of the $+$ operator in traditional DCOPs to the setting in which the constraint functions are partially ordered. In the general case, the semantics of the operator depends on that of the vector functions. To understand this, we consider again

the different examples of complex interactions which we are going to present in the next chapters of this thesis. When the different parameters defining such interactions represent multiple objectives (Chapter 5), then $\otimes$ is the element-wise sum of the different parameters. When the outcome of the constraint functions is uncertain and the vectors represent the mean and the variance associated with their pdfs (Chapter 6), then $\otimes$ is the convolution of these pdfs. Finally, when the outcome of the constraint functions is unknown (Chapter 7), then $\otimes$ is again the element-wise sum. However in this case it sums, respectively, the sample means and the bounds that are used to estimate such outcomes.

The operator $\max_{\succ}$ describes the optimisation criteria. In more detail, it takes as input sets of partially ordered instances $S = \{S_1, S_2, \ldots, S_m\}$ and outputs the best ones according to the order imposed by the binary relation $\succ$. Similarly to the $\otimes$ operator it is a generalisation of the max operator used for canonical DCOPs to the setting where the constraint functions in C are partially ordered. As discussed at the beginning of this chapter, this relation represents the key difference between canonical DCOPs and PO-DCOPs since it induces a partial order over the different vectors. According to this relation then, a single best value (i.e. a maximum or a minimum) might not exist anymore. Rather, multiple non-dominated values might exist which all need to be considered equally optimal. In general, vectors are not totally ordered. They are partially ordered according to a binary relation $\succ$ which is able to determine whether between two instances A and B, $A \succ B$ (A dominates B) holds, $B \succ A$ holds or A and B are not comparable.

This relation is then used by $\max_{\succ}$ to calculate the set of non-dominated alternatives— the instances that are *not* comparable with each other, but that dominate all the other instances— given the binary operator $\succ$. Similarly to the $\otimes$ operator, the semantics of $\max_{\succ}$ depends on the semantics of the constraint functions in the corresponding PO-DCOP. To understand this, consider a generic set of $S = \{A, B, C\}$. A relation $\succ$ exists between the different elements of this set such that the following ordering relations hold: $A \succ B$ and $B \succ C$. According to this, we can then define the best values of S as $\max_{\succ}(S) = \{A, B\}$, because we know that both A and B are better than C, but nothing can be said about which is better between them.

Having defined the operators, we can now define the set of non-dominated values corresponding to the solutions of a PO-DCOP as follows:

$$ND_X = \left\{ s^* \mid s^* = \max_{\substack{\succ \\ X}} \bigotimes_{j=1}^{m} C_j(X_j) \right\} \tag{4.2}$$

Similarly, the optimal solutions of a PO-DCOP are then defined as follows:

$$O_X = \left\{ X^* \mid X^* = \arg \max_{X \succ} \bigotimes_{j=1}^{m} C_j(X_j) \right\} \tag{4.3}$$

To calculate these solutions, we adopt the approach presented in Chapter 2 to define GDL-based algorithms for traditional DCOPs. In essence, we need to define the operators $\otimes$ and $\max_{\succ}$ for each PO-DCOP to instantiate the message passing algorithms 2.2 and 2.1. In more detail, in the next section we show how we generalise the GDL for deriving new algorithms for solving PO-DCOPs.

4.2 A Class of Algorithms for solving PO-DCOPs based on the GDL

We presented in Chapter 2 various complete DCOP algorithms building upon the GDL, such as DPOP, bounded max-sum and Action GDL. We discussed how these algorithms essentially proceed in three phases:

Phase 1 : This phase consists of encoding the DCOP into an acyclic factor graph. This can be done in two different ways, either by computing a spanning tree or by computing a junction tree. The former approach yields an approximate algorithm, whereas the second one yields an optimal one.

Phase 2 : This phase is the core of the algorithm, and uses the GDL message passing procedure (Algorithms 2.1 and 2.2) to solve the DCOP encoded in the previously defined factor graph.

Phase 3 : This phase is used to allow the agents to select a consistent variable assignment by using the value propagation algorithm.

The aim of this Section then is to use these three phases to derive algorithms to solve PO-DCOPs. In essence, as we have already discussed in Chapter 2, this consists, essentially, of re-defining phase 2 because the two other phases can use the same approaches used for canonical DCOPs (when using a spanning tree, we will need to re-define the weighting approach, as will be discussed in Chapter 5). In what follows we then re-define each of these phases to derive algorithms to solve PO-DCOPs.

4.2.1 The Factor Graph Computation Phase

This phase is the same as for canonical DCOP algorithms discussed in Chapter 2. In essence, it builds upon a key property of the GDL that whenever a factor graph encoding

a DCOP is acyclic then any GDL based algorithm is guaranteed to converge to an optimal solution (see Theorem 2.7). Thus, by manipulating factor graphs until they become acyclic, it is possible to provide quality guarantees on the solutions recovered. This manipulation happens in two different ways, the details of which are discussed in Chapter 2. Either a spanning tree algorithm is used to prune edges of the factor graph until it becomes acyclic or a junction tree algorithm is used to merge functions or variables until, again, the factor graph becomes acyclic. In so doing, the first approach yields a bounded approximate algorithm, whereas the second yields an optimal algorithm. In light of this, depending on the type of algorithm that needs to be instantiated, either optimal or approximate algorithms can be obtained. For the sake of our work we will use both these approaches, as will be explained with more precision in the next chapters.

4.2.2 The Message Passing Phase

In Chapter 2, we discussed how GDL algorithms exploit the fact that a DCOP's objective function is expressible using a *commutative semi-ring*, an algebraic structure $\langle R, \oplus, \otimes \rangle$, presenting a set of properties, among which the distributivity of the $\otimes$ operator over the $\oplus$ operator, which can be used to instantiate efficient algorithms. In essence, the former operators $\oplus$ and $\otimes$ are used to instantiate Algorithms 1 and 2 which solve Equation 2.11 by message passing between the vertices of the graph constructed in phase 1.

Hence, the key idea to solve a PO-DCOP is to define a commutative semi-ring characterising the objective function of the problem (Equation 4.3). We discussed in the previous section, how a PO-DCOP is defined in terms of two operators $\otimes$ and $\max_{\succ}$. The first aggregates vector functions, while the second is used to determine the non-dominated vectors. In particular, the $\max_{\succ}$ operator ranks partially ordered vector functions. In light of this we can define the semi-ring of a PO-DCOP, namely the partially ordered GDL semi-ring (PO-GDL) as follows:

Definition 4.2. The PO-GDL is a commutative semi-ring $\langle R, \max_{\succ}, \otimes \rangle$, such that:

- $R = \mathcal{P}(S)$ is the powerset (i.e. the set of all subsets) of S. Set S contains vectors whose elements take value in $\mathbb{R}$ [1].
- $\max_{\succ}$ takes sets of vectors $S_1, S_2, \ldots, S_m \subseteq R$ and outputs a set S' such that:

$$S' = \max_{\succ}\{S_1, S_2, \ldots, S_m\} = \left\{ V \in \bigcup_{i=1}^{m} S_i \mid \nexists V' \in \bigcup_{i=1}^{m} S_i : V' \succ V \right\} \tag{4.4}$$

  where the binary operator $\succ$ is a partial order relation.

[1] The identity elements **0** and **1** are defined depending on the specific instantiation of the PO-DCOP.

- $\otimes$ is a binary operator which aggregates sets of elements. More specifically, for any $S_1, S_2 \in R$:

$$S_1 \otimes S_2 = \{s_1 \otimes s_2 \mid \forall s_1 \in S_1, \forall s_2 \in S_2\} \tag{4.5}$$

The key idea of the PO-GDL semi-ring is that it is defined over *sets* of partially ordered instances rather than scalar values as for the canonical max-sum semi-ring used in DCOPs. Indeed, the $\max_{\succ}$ operator is used to rank sets of vectors of parameters representing partially ordered values of potential PO-DCOP's solutions. In more detail, the $\max_{\succ}$ operator is based on the $\succ$ operator, a partial order binary relation which we use to rank vector functions (see Section 4.1).

Now, as we will see with more detail in the following chapters, there exist problems for which the operator already exists in the literature. For example, in the next chapter, we will study multi-objective DCOPs, an instantiation of DCOPs involving maximising multiple conflicting objectives. In this case, the $\succ$ operator is the Pareto dominance defined in Chapter 2. However, there are also PO-DCOPs in which the $\succ$ operator needs to be derived accordingly. As we will see, this is the case when addressing DCOPs with uncertainty over the outcome of the agents' decisions. Within these settings, it becomes necessary to use a further function, whose aim is to correctly represent the partial order between the different DCOPs. Two examples, which we will consider in the remainder of this thesis, are the agents' risk profile and the collective upper confidence bound (UCB) which we will use to solve DCOPs involving uncertainty and in which the outcome of the agents' decisions is unknown (see Chapters 6 and 7).

The $\otimes$ operator is similar to the $\max_{\succ}$ operator. The key idea in this case is to aggregate sets of partially ordered vectors representing potential values of the solutions of a PO-DCOP. This is done in an element-wise fashion: all the vectors within each set are aggregated together. Note that, as we will see in Section 4.3, by doing so, the number of potential solutions of a PO-DCOP is greatly increased. As a consequence, the size of the messages used by the GDL message passing phase is increased as well.

4.2.3 The Value Propagation Phase

At the end of the second phase, each variable computes the set O_{x_i} of marginal non-dominated variable assignments for x_i, which is obtained by maximising over the marginal function Z_i (see Chapter 2 and Theorem 4.3):

$$O_{x_i} = \left\{ x_i^* \mid x_i^* = \arg \max_{x_i}{}_{\succ} Z_i(x_i) \right\}$$

If, for any variable x_i, $|O_{x_i}| > 1$, then there exists multiple solutions to the PO-DCOP, and a value propagation phase is needed to allow the agents to collectively select a consistent solution. However, in PO-DCOPs, each solution in O_{x_i} might result in multiple non-dominated values in ND_{x_i}, each corresponding to a different global assignment in X. For this reason, the canonical value propagation phase used for DCOPs does not work anymore, because now there are multiple optima and not a single one. Hence, we extend the canonical value propagation phase presented in Chapter 2, by having agents select an assignment $x_i^* \in O_{x_i}$. To select such a solution, value propagation proceeds by passing messages between the variables and functions of the acyclic factor graph. First, the variable x_r with the lowest index is chosen as the *root* of the tree, and is responsible for initiating the value propagation phase by selecting a non-dominated assignment x_r^*. Second, the variable sends value-propagation messages ($x_r = x_r^*$) to all the function nodes to which the variable is connected.

The behaviour of all the other nodes in the graph then depends on their type. More specifically:

Function nodes ($C_j \in C$) : Upon receiving a message ($x_i = x_i^*$) from variable x_i, the constraint function C_j computes the set $O(X_j \setminus \{x_i\})$ of marginal local optimal solutions for variables $X_j \setminus \{x_i\}$, conditioned on $x_i = x_i^*$. Next, value propagation selects one optimal assignment $x_k^* \in O(X_j \setminus \{x_i\})$ for each variable $x_k \in X_j \setminus \{x_i\}$ ($k \neq j$) and sends the message ($x_k = x_k^*$) to x_k.

Variable nodes ($x_i \in X$) : For each non-root variable x_i, once it receives a message ($x_i = x_i^*$) from a function C_j, it sets its value to x_i^* and propagates the message ($x_i = x_i^*$) to all the function nodes C_k, $k \in \mathcal{M}(i)$.

Note that, during value propagation, a single message is sent across each edge of the factor graph. Thus, the algorithm terminates once each non-root variable has received a value-propagation message.

4.3 Theoretical Analysis

We analyse in this section two key theoretical properties of PO-GDL algorithms. First, we demonstrate the optimality of the PO-GDL message passing phase over acyclic factor graphs. This property, as mentioned at the beginning of this chapter, is fundamental to define algorithms capable of providing quality guarantees, one of the key requirements of our work (Chapter 1). The reason is that this property guarantees that the second phase of our algorithm will *always* retrieve the optimal solutions of the graph produced at the end of phase one. Second, we study the complexity of PO-GDL algorithms and show how

solving PO-DCOPs is, in general, more complex than solving traditional DCOPs since now multiple parameters and multiple non-dominated solutions need to be considered.

4.3.1 Optimality of the PO-GDL Phase

Canonical DCOPs can be represented using factor graphs, a special type of bipartite graph in which vertices can represent either variables in X or constraint functions in C. In light of this, factor graphs can also be used to represent PO-DCOPs. Knowing this, we need to demonstrate that the PO-GDL possesses the same fundamental property as the canonical GDL message passing phase, to use it to derive algorithms for solving PO-DCOPs. This means that we need to show that it computes *all* the non-dominated solutions of a PO-DCOP whose corresponding factor graph is *acyclic*. In more detail, we show that the following theorem holds:

Theorem 4.3. *Given a PO-DCOP $\langle A, X, D, F, C\rangle$, if the problem's corresponding factor graph is acyclic and the stopping criterion is chosen such that the algorithm is run for a number of iterations equal to the diameter of the factor graph, the following equation holds for each variable $x_i \in X$:*

$$Z_i(x_i) = \max_{\substack{\succ \\ X\setminus x_i}} \bigotimes_{j=1}^{m} C_j(X_j) \tag{4.6}$$

Proof. The proof follows directly from Theorem 2.7 and from the fact that the objective function of PO-DCOPs can be characterised as a commutative semi-ring as shown in Section 4.2. More formally, $\langle R, \max_{\succ}, \otimes\rangle$ is a commutative semi-ring (Section 4.2). This means that $\max_{\succ}$ distributes over $\otimes$. Hence, by replacing, the operator $\oplus$ in Algorithms 2.1 and 2.2 with operator $\max_{\succ}$ it follows that Equation 4.6 generalises Equation 2.12 to PO-DCOPs and thus, that the theorem holds. □

Note that, due to theorem 4.3, function $Z_i(x_i)$ is now a function in which each assignment x_i^* can yield multiple non-dominated values, $Z_i(x_i^*) = \{v^* \mid \forall v \text{ s.t. } v^* = \max_{\succ X\setminus x_i} \bigotimes_{j=1}^{m} C_j(X_j)\}$. Each of these values corresponds to a solution of a PO-DCOP for which variable x_i is assigned to value x_i^*. This means that at the end of the message passing phase each agent retrieves its multiple optimal values, for which value propagation is necessary to obtain a consistent assignment.

Now, to properly understand PO-DCOPs and the solution techniques that we have derived, it is important to understand what achieving optimality involves in terms of computation and communication complexity. This is done in the next section.

4.3.2 Complexity

We can derive the computation and communication complexity of PO-GDL algorithms by using properties inherited from canonical GDL algorithms. In particular, any PO-GDL algorithm exploits the factorisability of the problems it is solving. Therefore the scope of each constraint function $C_j(X_j)$ contains only the variables associated with the constraint.

In light of this, computing messages $R_{j\to i}(x_i)$ from function C_j to variable x_i (Algorithm 2) for any PO-GDL algorithms requires $O(|D_{max}|^{|X_j|})$ evaluations of function C_j, where D_{max} is the largest domain among variables X_j. This means that the computation is exponential *only* in the number of variables in the scope of C_j, not the total number of variables. This drastically reduces the computation required by each agent if compared with a centralised technique where each of them need to consider all the variables in X.

Now, in the worst case scenario, each possible variable assignment of a PO-DCOP is an optimal solution. Formally, this means that any variable assignment X' of the variables in X is an optimal assignment (i.e. a solution of the PO-DCOP) in O_X. In terms of computation, this means that all the non-dominated values in ND_X, corresponding to each optimal solution in O_X, are going to be propagated by the message passing phase.

Hence, assuming that each constraint function is composed of k parameters and that a sufficient (but finite) number of messages have been exchanged (as discussed in Chapter 2), this implies that these messages may contain up to $O(k \times |D_{max}|^{n+1})$ non-dominated values. To understand the way this size is calculated, consider that in the worst case scenario each possible variable assignment is a solution of the problem and corresponds to a set of non-dominated solutions. Hence, in the worst case, there exists $|D_max|$ sets of non-dominated vectors corresponding to a PO-DCOP solution. Each variable's assignment then corresponds to a non-dominated solution. Since there are n variables, each of these sets may contain at most D^n_{max} non-dominated vectors of size k, which means that the size of the messages is at most $k \times |D_{max}| \times |D_{max}|^n = k \times |D_{max}|^{n+1}$.

Thus, this means that the size of the messages is linear in the number of parameters k and exponential in the number of variables n. This is a linear increase (of a factor k) with respect to a traditional DCOP. Hence, the factor k does not influence much the size of the messages compared to the factor n, which is very likely to be quite large in a PO-DCOP and which is the real constraint for using our algorithms in a real world domain. Nonetheless, this is to be expected, since we are now trying to solve a problem more complex that a canonical DCOP. Moreover, as stated in Chapter 1, in this thesis we are focusing principally on the theory behind PO-DCOPs and their solution techniques. In our future work, as we will discuss in Chapter 8, we plan to investigate the way to adapt these problems to real world settings. Finally, it should be noted that $\prod_{i=1}^{n} |D_i|$

is an upper bound of the number of possible solutions, which is equal to the size of the Cartesian product of the domains of all variables.

4.4 Summary

This chapter presented a novel problem framework and a class of solution techniques to solve coordination problems involving complex agent interactions. In so doing, we proposed a novel class of problems and solutions capable of addressing the requirements of generality and complex interactions presented in Chapter 1. More specifically, we presented in Section 4.1 the class of PO-DCOPs which generalises the canonical DCOP framework by re-defining the constraint functions as partially ordered constraint functions represented as vector functions. The key idea is that the vectors represent the set of parameters necessary to define the complex interactions of the different agents.

In Section 4.2, we then presented the PO-GDL, a class of algorithms that exploits the properties of the GDL framework described in Chapter 2 to solve PO-DCOPs. The key idea is to define the commutative semi-ring characterising the objective function of a PO-DCOP and then use it to instantiate the message passing algorithms specific to the GDL (Algorithms 2.1 and 2.2). In so doing, we can derive algorithms capable of meeting the performance guarantee requirement discussed in Chapter 1.

However, as we discussed in Section 4.3, the complexity of the problems may be prohibitive for problems where the number of agents and the number of parameters characterising their interactions are large. Thus, the requirements of scalability and robustness may only be partially met depending on the specific problem instance. To confirm this hypothesis, in the remaining chapters of this thesis, we focus on studying three different types of PO-DCOPs in which the complex interactions involve multiple objectives (Chapter 5), risk and uncertainty (Chapter 6) and online learning (Chapter 7).

Chapter 5

A Class Of Algorithms for Partially Ordered Coordination Problems involving Multiple Objectives

This chapter presents a formal framework to solve partially ordered distributed constraint optimisation problems (PO-DCOPs) involving multiple objectives and proposes a class of algorithms to solve these problems. In more detail, we specialise the partially ordered DCOP framework (PO-DCOP) and the corresponding class of algorithms based on the generalised distributive law (GDL), the partially ordered GDL (PO-GDL), to solve DCOPs involving multiple objectives (MO-DCOPs). In so doing, the aim of this chapter is to address the requirements of complex interactions involving multiple objectives discussed in Chapter 1.

In more detail, we discussed in Chapter 1 that many real world coordination problems involve the simultaneous optimisation of multiple and possibly conflicting objectives. For instance, referring back to the scenario described in Chapter 1, in disaster response, unmanned aerial vehicles (UAVs) are deployed to complete imagery collection tasks. One of their objectives is to maximise the number of completed tasks. However, some of the areas where imagery needs to be collected might contain hazards for the agents, such as fire, water or obstacles. Hence, to prevent the loss of the vehicles, the agents might consider minimising a second objective, the number of completed dangerous tasks. However, the optimisation of one objective may result in detrimental performance in terms of the other because the agents may not complete dangerous tasks, and thus the number of completed tasks will most likely decrease. Now, trading off between such conflicting

objectives induces a partial order over the solutions of the coordination problems. Unfortunately, as discussed in the previous chapter, measuring the quality of these partially ordered solutions cannot be done using canonical real valued functions. For this reason, traditional DCOPs and GDL algorithms cannot be used and new problems and solutions techniques need to be derived.

Against this background, this chapter produces two contributions. We first propose the MO-DCOP framework which specialises PO-DCOPs to address multiple objectives (Section 5.1). To achieve this, we re-define the constraint functions of a PO-DCOP (see Chapter 4) as multiple objective functions. As a consequence, the solutions of the corresponding global function are partially ordered according to the Pareto dominance relation (Chapter 2). In so doing, we provide a novel generalisation of the canonical DCOP framework which allows us to encompass a broader set of optimisation problems. Then, in Section 5.2, we propose a new class of algorithms for solving MO-DCOPs. More specifically, we exploit the PO-GDL method described in Chapter 4 to instantiate a novel message passing procedure based on a novel semi-ring, namely the Pareto dominated / sum (PDS) semi-ring, which we derive to transform any GDL-based algorithm into an algorithm for solving MO-DCOPs. In so doing, we bring multiple canonical DCOP algorithms such as Action GDL, bounded max-sum and DPOP, whose level of performance has been well studied in literature, to the multi-objective domain.

Next, since the number of Pareto optimal solutions can be large, we instantiate an equivalent of the bounded max-sum algorithm to solve MO-DCOPs and evaluate how the increase in the number of Pareto optimal solutions affects the performance of the algorithm. In so doing, we can study how partially ordered solutions will affect the performance of PO-GDL algorithms. Thus, we empirically evaluate our algorithm, which we name bounded multi-objective max-sum (B-MOMS), by benchmarking it against a centralised optimal algorithm on a multi-objective extension of the graph colouring problem, a standard test problem for DCOP algorithms (Modi et al., 2005). Our results show that the approximation ratio never exceeds 2 (i.e. the solution's quality is never more than 50% away from the optimal), even for extremely constrained problems (i.e. fully connected graphs), and is less than 1.5 for graphs where constraints exist between 20% of all pairs of agents for 14 variables. In terms of the requirements defined in Section 1, it should be noted that the complexity of the problem is increased since the algorithm now has to keep track of multiple Pareto optimal solutions. As a consequence, the requirements of generality, robustness, scalability and resource-awareness are not satisfied to the same extent as they would be in the single-objective case. Nonetheless, the results indicate that the runtime required by B-MOMS never exceeds 30 minutes, even for maximally constrained graphs with 100 agents, positioning it well within the confines of many real-life applications. In addition, the empirical evaluation shows that

the algorithm is able to provide performance guarantees even in extremely complex problems (maximally constrained graphs).

The remainder of this chapter proceeds as follows: Section 5.1 describes the general MO-DCOP framework, Section 5.2 illustrates B-MOMS a bounded approximate algorithm that builds upon the PO-GDL to solve MO-DCOPs, Section 5.3 presents some empirical evaluation of the algorithm and finally, Section 5.4 summarises the chapter.

5.1 Problem Definition

This section describes the framework of multi-objective DCOPs (MO-DCOPs), a generalisation of canonical DCOPs to incorporate multiple and possibly conflicting objectives. In essence, a MO-DCOP can be seen as the problem of maximising simultaneously k different DCOPs defined over the same variables, and whose constraint functions might conflict with one another. More formally, a MO-DCOP is defined as follows:

Definition 5.1. A MO-DCOP is a tuple $\langle A, X, D, F, C \rangle$ such that:

- $A = \{1, \ldots, |A|\}$ is a set of agents.
- $X = \{x_1, x_2, \ldots, x_n\}$ is a set of variables.
- $D = \{D_1, \ldots, D_n\}$ is a set of discrete domains, where D_i is the domain of variable x_i.
- $F : X \to A$ is a function that assigns variables to agents. Each agent controls (the value of) the variables that are assigned to it.
- $C = \{C_1, \ldots, C_m\}$ is a now set of multi-objective (instead of deterministic scalar) constraint functions. Hence, each $C_j \in C$ is now a vector of objective functions, defined over the set of variables X_j:

$$C_j(X_j) = \left[C_j^1(X_j), \ldots, C_j^k(X_j)\right]^T \tag{5.1}$$

  where k is the number of objectives of the problem.

Thus, in a MO-DCOP, as is the case for a general PO-DCOP, the sets A, X, D and F maintain the same semantics as for traditional DCOPs. The constraint functions C_j are then generalised into multi-objective functions. This means that their assignments $C_j(X_j)$ are partially ordered. Thus, there is no longer necessarily a single optimal solution X^* as for canonical DCOPs. Rather, as discussed in Chapter 2, we now have a

set of Pareto optimal solutions, corresponding to a set of non-dominated values. More formally, we define the set of non-dominated values of a MO-DCOP as:

$$ND_X = \left\{ v \mid v = \max_{\succ_X} \sum_{j=1}^{m} C_j(X_j) \right\} \tag{5.2}$$

Equation 5.2 is a specialisation of Equation 4.2 (see Chapter 4) to the multi-objective domain. Similarly, we can specialise Equation 4.3 (Chapter 4) to define the set of Pareto optimal (PO) solutions of a MO-DCOP as:

$$\begin{aligned} PO_X &= \left\{ X^* \mid X^* = \arg\max_{\succ_X} \sum_{j=1}^{m} C_j(X_j) \right\} \\ &= \left\{ X^* \mid X^* = \arg\max_{\succ_X} \sum_{j=1}^{m} \left[C_j^1(X_j), C_j^2(X_j), \ldots, C_j^k(X_j) \right]^T \right\} \end{aligned} \tag{5.3}$$

The solutions $X^* \in PO$ can be characterised as the set of Pareto optimal assignments that optimise the sum of the partially ordered constraint functions. These correspond to the set of non-dominated values ND that "maximise" Equation 5.3, following the dominance operator $\succ$ defined in Chapter 4. In light of this, MO-DCOPs are PO-DCOPs (see Chapter 4) in which the arity of the vector functions is reduced to 1. Hence, the following property holds:

Property. A MO-DCOP is a PO-DCOP in which each partially ordered constraint function $C_j \in C$ represents a multi-objective function defined over k objectives.

In light of this, we can define the constraint function of a MO-DCOP as a commutative semi-ring (see Chapters 2 and 4) and use it for deriving new algorithms. In more detail, Equation 5.2 is essentially a sum of constraint functions encoding multi-objective vectors. These vectors are aggregated using the element-wise addition, while the notion of Pareto dominance is used to discriminate between the different solutions. We can then use these properties to define the Pareto-dominated / sum semi-ring (PDS), as follows:

Definition 5.2. PDS is a commutative semi-ring $\langle R, \max_{\succ}, + \rangle$, such that:

- $R = \mathcal{P}(S)$ is the powerset (i.e. the set of all subsets) of S. Set S contains all multi-objective vectors where each element takes value in $\mathbb{R}$[1].

[1]The identity elements **0** and **1** are singleton sets with a vector in which each element is a 0 and a vector in which each element is a 1.

- $\max_{\succ}$ takes sets of multi-objective vectors $S_1, S_2, \ldots, S_m \subseteq R$ and outputs a set S' such that:

$$\begin{aligned} S' &= \max_{\succ}\{S_1, S_2, \ldots, S_m\} \\ &= \left\{ V \in \bigcup_{i=1}^{m} S_i \mid \nexists V' \in \bigcup_{i=1}^{m} S_i : V' \succ V \right\} \end{aligned} \tag{5.4}$$

  where the binary operator $\succ$ is a partial order relation (see Chapter 4) based on the Pareto dominance.

- $+$ is a binary operator such that if $S_1, S_2 \in R$:

$$S_1 + S_2 = \{\forall i \in (0, \ldots, k)\ V_1(i) + V_2(i) \mid \forall V_1 \in S_1, \forall V_2 \in S_2\} \tag{5.5}$$

Given this semi-ring, one can easily verify that the distributive law applies and thus that $\max_{\succ}$ distributes over $\otimes$. Hence, the complexity of optimising the global constraint function (Equation 5.2) can be factorised into simpler components.

In general, a MO-DCOP allows us to represent a coordination problem in which the agents make decisions over multiple, conflicting objectives. We consider here the case where the agents (either human or software) have no preferences between the objectives. Contrarily, as discussed in Chapter 2, it would be possible to cast the problem as a standard optimisation problem where the objective function is a weighted sum of the different objectives and where each weight encodes the agents preferences.

In addition, we consider here the setting where the objectives are conflicting. Indeed, such objectives are a feature of many real world problems (see Chapter 2 for some examples). In fact, if they were independent, they could be solved using traditional DCOPs techniques. To understand the notion of conflicting objective, consider again the domain of disaster response. One of the key scenarios discussed in Chapter 2 involves unmanned aerial vehicles (UAVs) deployed to search lost life-rafts while tracking down the discovered ones. The two objectives could be modelled as two global constraint functions C_s^1 and C_t^1, one for searching and one for tracking, which are then decomposed into a sum of agents interactions as described in Chapter 3. These functions are then two distinct objectives of the problem. In addition, they conflict since assigning more UAVs to track life-rafts (i.e. focusing on the "tracking" objective) implies that less UAVs will be used for searching (i.e. the "search" objective is neglected). Hence, in this setting, the optimisation of one objective will result in detrimental performance in terms of the other. The following example illustrates a possible MO-DCOP related to disaster response:

Example 5.1. *Consider a MO-DCOP with* $A = \{A_1, A_2\}$, $X = \{x_1, x_2\}$, $D = \{\{0, 1\}, \{0, 1\}\}$, *and* $C = \{C_1(x_1, x_2), C_2(x_1, x_2)\}$ *(Figure 5.1 illustrates the factor graph encoding this DCOP). Constraint functions* C *are shown in the following table:*

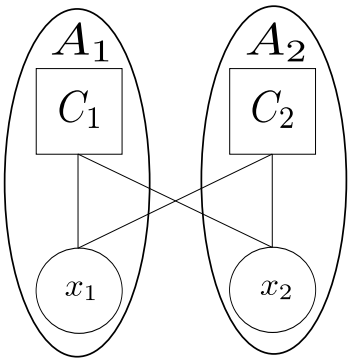


FIGURE 5.1: A factor graph encoding the DCOP presented in Example 5.1.

x_1	x_2	$C_1(x_1, x_2)$	$C_2(x_1, x_2)$	$C(x_1, x_2)$
0	0	$[3, 1]$	$[3, 2]$	$[6, 3]$
0	1	$[1, 2]$	$[1, 1]$	$[2, 3]$
1	0	$[2, 0]$	$[2, 4]$	$[4, 4]$
1	1	$[2, 1]$	$[0, 1]$	$[2, 2]$

In this case the set of non-dominated values is defined as:

$$ND_X = \{[6, 3], [4, 4]\}$$

These values correspond to the following set of Pareto optimal solutions:

$$PO_X = \{\{0, 0\}, \{1, 0\}\}$$

Moreover, the utopia point (Definition 2.4.3.1) is defined as follows:

$$V^* = [6, 4]$$

As shown in the example, solving a MO-DCOP requires trading off between different objectives and as a consequence it is non-trivial. The use of standard DCOP algorithms is not advocated in this case, since it is likely to result in very poor performance (Marler and Arora, 2004). Indeed, employing such approaches would require the use of scalarisation techniques which consist of weighting the various objective functions and incorporate them into a single global real valued function which could then be solved using canonical techniques (Chapter 2). However, it is well known that using this type of approach is likely to yield poor solutions far from the optimal (Marler and Arora, 2004). For this reason, in the next section we present a class of algorithms that were specifically derived to solve these problems.

5.2 An Algorithm for solving MO-DCOPs based on the PO-GDL

We discussed in Chapter 4, how the objective function of a PO-DCOP possesses a specific algebraic structure (i.e. a commutative semi ring) which can be used to derive various decentralised algorithms to solve it, all based on the GDL framework. In the same chapter, we also discussed the way these algorithms can be instantiated by following the methodology used to derive canonical DCOP algorithms, such as DPOP, Action GDL and bounded max-sum (see Chapter 2).

In this chapter, we are interested in solving MO-DCOPs an instantiation of PO-DCOPs in which the agents interact over multiple objectives. To solve these problems, this section then proposes a generalisation of the bounded max-sum algorithm, a well known bounded approximate algorithm to solve canonical DCOP discussed in Chapter 2. Given the inherent complexity of MO-DCOPs, the choice of an approximate algorithm seems the better option. In addition, this choice allows to compute effective bounds on the quality of the recovered solutions even in the case of very complex problems. In more detail, we propose an algorithm proceeding over three phases, as follows:

Phase 1 - Bounding : The *bounding* (B) phase extends the approach used for the bounded max-sum algorithms to the multi-objective domain. In so doing, this phase produces an acyclic factor graph encoding a bounded approximation of the original MO-DCOP.

Phase 2 - Message Passing : The *message passing* (MP) phase is a re-definition of the PO-GDL message passing phase allowing the agents to coordinate to find the Pareto optimal set of solutions to the cycle-free factor graph computed in the bounding phase.

Phase 3 - Value Propagation : The *value-propagation* (VP) phase allows the agents to select a consistent variable assignment, considering that now multiple non-commensurable alternatives exist.

In what follows, we describe each of these phases in detail, and then we analyse some of the algorithm's key properties.

5.2.1 Algorithm Description

Our algorithm proceeds in three phases. In what follows, we discuss each of these phases in turn.

5.2.1.1 The Bounding Phase

This phase calculates the bounds on the solutions of our algorithm. In more detail, we propose a novel weighting approach that generalises the one described in Chapter 2. The key idea is to generalise the edge weights w_{ji}, used to bound the max-sum algorithm, from scalar to vector values. The key issue is then how to use the weighting approach used for max-sum on each of the different objectives of the MO-DCOP. To achieve this, we first compute the impact of each variable x_i in the scope of each local multi-objective constraint function C_j over all the objectives of the problems (we assume there are k objectives):

$$\mathbf{w}_{ji} = \left[w_{ji}^1, \ldots, w_{ji}^k\right]^T$$

Each scalar weight w_{ij}^o $(1 \leq o \leq k)$ is defined as follows:

$$w_{ji}^o = \max_{X_j \setminus \{x_i\}} \left[\max_{x_i} C_j^o(X_j) - \min_{x_i} C_j^o(X_j)\right]$$

Since the problem of finding a maximum spanning tree is defined on instances with scalar edge weights (Rogers et al., 2011), it is necessary to rank the vector weights. This procedure must ensure that the resulting ordering favours deletion of dominated vectors over non-dominated ones. One way of doing this is to assign a scalar weight w_{ji} to each vector $\mathbf{w}_{ji}$, which is proportional to the number of edge weights it dominates. More formally:

$$w_{ji} = -|\{\mathbf{w}_{nm} \mid \mathbf{w}_{ji} \preceq \mathbf{w}_{nm}, (j,i) \neq (n,m)\}| \tag{5.6}$$

Thus, using this scalarisation, non-dominated weight vectors are assigned a value of 0, vectors dominated by a single element are assigned a value of -1, and so on. With these scalar edge weights, the same spanning tree algorithm used for the canonical bounded max-sum can be used. The following example discusses how the bounding phase is applied to Example 5.1:

Example 5.2. *The bounding phase starts with the two functions C_1 and C_2 computing their vector weights as detailed above. Specifically, C_1 computes:*

$$w_{11}^1 = \max_{x_2}[\max_{x_1} C_1^1(x_1, x_2) - \min_{x_1} C_1^1(x_1, x_2)] = 1$$
$$w_{11}^2 = \max_{x_2}[\max_{x_1} C_1^2(x_1, x_2) - \min_{x_1} C_1^2(x_1, x_2)] = 1$$

where w_{11}^1 *and* w_{11}^2 *are the two weights related to the edge between* C_1 *and* x_1*. Moreover, the factor computes:*

$$w_{12}^1 = \max_{x_1}[\max_{x_2} C_1^1(x_1, x_2) - \min_{x_2} C_1^1(x_1, x_2)] = 2$$
$$w_{12}^2 = \max_{x_1}[\max_{x_2} C_1^2(x_1, x_2) - \min_{x_2} C_1^2(x_1, x_2)] = 1$$

where w_{12}^1 *and* w_{12}^2 *are the two weights related to the edge between* C_1 *and* x_2*. In a similar fashion* C_2 *computes:*

$$w_{21}^1 = \max_{x_2}[\max_{x_1} C_2^1(x_1, x_2) - \min_{x_1} C_2^1(x_1, x_2)] = 1$$
$$w_{21}^2 = \max_{x_2}[\max_{x_1} C_2^2(x_1, x_2) - \min_{x_1} C_2^2(x_1, x_2)] = 2$$

where w_{21}^1 *and* w_{21}^2 *are the two weights related to the edge between* C_2 *and* x_1*. Finally, the factor computes:*

$$w_{22}^1 = \max_{x_1}[\max_{x_2} C_2^1(x_1, x_2) - \min_{x_2} C_2^1(x_1, x_2)] = 2$$
$$w_{22}^2 = \max_{x_1}[\max_{x_2} C_2^2(x_1, x_2) - \min_{x_2} C_2^2(x_1, x_2)] = 3$$

where w_{22}^1 *and* w_{22}^2 *are the two weights related to the edge between* C_2 *and* x_2*. The resulting weighted factor graph is shown in Figure 5.2. After the weights are computed, the agents share these weights in order to compute the different scalar vectors. More specifically, the following weights are computed, using Equation 5.6:*

$$\begin{aligned} w_{11} &= -3 \\ w_{12} &= -1 \\ w_{21} &= -1 \\ w_{22} &= 0 \end{aligned} \tag{5.7}$$

Thus, since w_{11} *is the smallest weight on the novel weighted factor graph, it is pruned from the graph by the spanning tree algorithm. The set of values, corresponding to the cycle free factor graph is shown in Table 5.1. Moreover, in this specific example, the bound to the utopia point can be easily calculated as* $\mathbf{W} = \mathbf{w}_{11} = [1, 1]$*. The resulting acyclic factor graph, shown in Figure 5.2 constitutes the input of the second phase of the algorithm, which follows.*

At the end of the bounding phase, an acyclic factor graph representing a maximum spanning tree is produced. This factor graph encodes a bounded approximation of the original MO-DCOP to be solved. Since it is acyclic, it can be optimally solved by using a GDL message passing phase, which we discuss next.

x_1	x_2	$\min_{x_2^c} C_1(x_1, x_2^c)$	$C_2(x_1, x_2)$	$C(x_1, x_2)$
0	0	$[2,0],[0,1]$	$[3,2]$	$[5,2],[3,3]$
0	1	$[2,0],[0,1]$	$[1,1]$	$[2,1],[1,2]$
1	0	$[1,0]$	$[2,4]$	$[3,3]$
1	1	$[1,0]$	$[0,1]$	$[1,1]$

TABLE 5.1: The resulting bi-objective function after pruning the factor graph in Figure 5.2. Here x_2^c is the variable whose corresponding edge has been pruned from the factor graph.

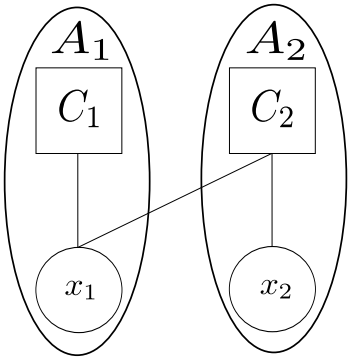


FIGURE 5.2: The pruned factor graph produced by the bounding phase of the B-MOMS algorithm described in Example 5.2.

5.2.1.2 The Message Passing Phase

The message passing phase presented here works in the same way as those discussed in Chapters 2 and 4. The key idea is to specialise the PO-GDL by using the semi-ring $\langle R, \max_{\succ}, + \rangle$ defined in Section 5.1. By so doing, we can then use the operators $+$ and $\max_{\succ}$ to instantiate the message passing algorithms of the GDL framework (Algorithms 2.1 and 2.2) for solving MO-DCOPs. Moreover, the resulting message passing phase can be used to optimally solve any MO-DCOP whose resulting factor graph is acyclic (see Chapter 4 and Section 5.2.2) and, as a consequence, the one produced in the bounding phase. The following example illustrates the message passing phase related to Examples 5.1 and 5.2:

Example 5.3. *In order to keep the exposition as clear as possible, we consider a simple message passing schedule, where the computation starts at the leaf nodes and goes up through the different nodes of the graph. The computation begins at time step $t = 0$, nodes C_1 and x_2 both have a single edge (i.e. they are leaves of the cycle free factor graph) and therefore send a message first:*

$Q_{2\to 2}(x_2) =$

x_2	C
0	$[0,0]$
1	$[0,0]$

$R_{1\to 1}(x_1) =$

x_1	C
0	$[2,0],[0,1]$
1	$[1,0]$

Here x_2 sends a "zero" message because it has not received any information, while C_1 computes the multi-objective marginal function of variable x_1, as detailed above and in Chapter 4. At time step $t = 1$, x_1 and C_2 receive the messages and compute the next two messages:

$$R_{2\to1}(x_1) = \begin{array}{c|c} x_1 & C \\ \hline 0 & [3,2] \\ 1 & [2,4] \end{array} \qquad Q_{1\to2}(x_1) = \begin{array}{c|c} x_1 & C \\ \hline 0 & [2,0],[0,1] \\ 1 & [1,0] \end{array}$$

At time step $t = 2$, the final two messages are computed. Note that, while x_1 has received messages from all the previous variables and can therefore calculate its own optimal solutions, x_2 still requires information about C_1 (i.e. since the original edge has been pruned from the graph). Thus, x_1 and function C_2 compute the following messages:

$$Q_{1\to1}(x_1) = \begin{array}{c|c} x_1 & C \\ \hline 0 & [3,2] \\ 1 & [2,4] \end{array} \qquad R_{2\to2}(x_2) = \begin{array}{c|c} x_2 & C \\ \hline 0 & [5,2],[3,3] \\ 1 & [3,3] \end{array}$$

After these messages have been sent, the MS phase terminates. Finally, both the variables compute their marginal Pareto optimal solutions:

$$x_1^* = PO\left\{\begin{array}{c|c} x_1 & C \\ \hline 0 & [5,2],[3,3] \\ 1 & [3,3] \end{array}\right\}$$
$$= \{0,1\}$$

$$x_2^* = PO\left\{\begin{array}{c|c} x_2 & C \\ \hline 0 & [5,2],[3,3] \\ 1 & [2,1],[1,2] \end{array}\right\}$$
$$= \{0\}$$

Such solutions correspond to the two Pareto optimal assignments of the problem $PO(X) = \{(0,0),(1,0)\}$. The value propagation phase proceeds next.

Similar to canonical DCOP algorithms, it can happen that different solutions yield the same value of the global constraint function. This is even more likely for MO-DCOP

where there might exist numerous non comparable solutions which are all Pareto optimal. For this reason, a value propagation phase is required, which we discuss next.

5.2.1.3 The Value-Propagation Phase

The value-propagation phase operates on the cycle-free factor graph computed by the bounding approach. In this phase, variables and function nodes in the factor graph coordinate to select a consistent variable assignment. Since MO-DCOPs are PO-DCOPs, we can use the value propagation described in Chapter 4, by instantiating the $\succ$ relation as the Pareto Dominance as described in the previous section.

During value propagation, a single message is sent across each link in the factor graph. The algorithm terminates once each non-root variable has received a value-propagation message. The following example illustrates the value propagation phase related to the MO-DCOP described in Example 5.1, and after the bounding (Example 5.2) and the message passing (Example 5.3) phases have terminated:

Example 5.4. *The value-propagation phase starts right after the message passing phase (Example 5.3). At time step* 3*, variable* x_2*, as the pre-determined root of the graph, selects a specific assignment*[2] *and sends a* VP *message containing the selected value* $x_2 = 0$ *to* C_2*. At time step* 4*,* C_2 *receives the message, calculates the set of Pareto optimal solutions for which* $x_2 = 0$*, that is* $PO(X_2) = \{[x_1 = 0, x_2 = 0], [x_1 = 1, x_2 = 0]\}$*, selects one of the different alternatives*[3] *and sends a new* VP *message to* x_1 *containing* $x_1 = 1$*. Finally, at the time steps* 5 *and* 6*,* x_1 *receives the message, forwards it to* C_1*, and the computation terminates.*

Thus, the algorithm recovers the solution $X^* = \{x_1 = 1, x_2 = 0\}$*, which is, as can be seen in the table of Example 5.1, one of the Pareto optimal solutions of the problem, in this specific case.*

5.2.2 Theoretical Analysis

We now discuss some of the fundamental properties of our GDL algorithms based on the PDS semi-ring (see Definition 5.2).

[2] To keep the exposition as clear as possible, we assume that the selection criterion is to pick a random value. Other possibilities include selecting the variable with the lowest index or the agent with the lowest ID

[3] Again, we assume a random selection.

5.2.2.1 Optimality of the Message Passing Phase

In order to use the PDS to derive new algorithms, as described in Chapter 4, the following theorem must hold:

Theorem 5.3. *Given a MO-DCOP $\langle A, X, D, F, C\rangle$, if the problem's corresponding factor graph is acyclic and the stopping criterion is chosen such that the algorithm is run for a number of iterations equal to the diameter of the factor graph, the following equation holds for each variable $x_i \in X$:*

$$Z_i(x_i) = \max_{\substack{\succ \\ X\setminus x_i}} \sum_{j=1}^{m} C_j(X_j) \tag{5.8}$$

Proof. The proof follows from Theorem 4.3 and from the fact that the objective function of a MO-DCOP is defined over the PDS semi-ring, which is a commutative semi-ring. Thus, by instantiating the operator $\otimes$ of Algorithms 1 and 2, with the operator $+$ and by defining the dominance relation $\succ$ as the Pareto dominance relation, it follows that Equation 5.8 specialises Equation 4.6 to MO-DCOPs and thus, that the theorem holds. □

5.2.2.2 Bound on the Solutions Recovered

Having demonstrated the optimality of the message passing phase, we can now derive the bounds computed by the algorithm. In essence, we generalise the bounds computed by the bounded max-sum algorithm (Rogers et al., 2011) to the multi-objective case. To do so, first, we define vector $\mathbf{W} = [W^1, \ldots, W^k]$ as the sum of vector weights $\mathbf{w}_{ij}$ of the edges between C_j and x_i that were pruned in the bounding phase to obtain an acyclic graph (Section 5.2.1.1). To characterise the upper bound computed by the algorithm, we use the concept of *utopia point* (see Chapter 2), which in the multi-objective domain becomes an upper bound on the value of a Pareto optimal solution of a MO-DCOP. Hence, we can state the following theorem:

Theorem 5.4. *Given an arbitrary MO-DCOP, for any assignment $X' \in PO(X)$ computed by B-MOMS, the following bound holds:*

$$\sum C_j(X') + \mathbf{W} \geq \mathbf{V}^* \tag{5.9}$$

Proof. The theorem follows directly from the fact that we extend the bounded max-sum algorithm for single objective DCOPs to MO-DCOPs. In more detail, for each objective o ($1 \leq o \leq k$) of an MO-DCOP, by using the approach defined in (Rogers et al., 2011),

it is easy to see that the following bound holds:

$$C^o(X') + W^o \geq \max_{X'} C^o(X')$$

thus concluding the proof. □

Similarly, we define the problem dependent approximation ratio $\boldsymbol{\rho} = [\rho_1, \ldots, \rho_k]$ of the solutions computed by B-MOMS, where each ρ_i is given by Equation 2.9.

5.3 Empirical Evaluation

We present in this section some empirical evaluation in order to ascertain the performance of the B-MOMS algorithm that we have derived. To this end, we benchmark its performance against an optimal algorithm which consists of enumerating all the possible solutions of a MO-DCOP to find the Pareto optimal ones. In the remainder of this section, we present a multi-objective extension to the canonical graph colouring problem used in our experiments, detail the experimental setup, and discuss the results.

5.3.1 Multi-Objective Graph Colouring

In order to evaluate the performance of our algorithm, we consider a multi-objective extension of the graph-colouring problem, which is a well known benchmark problem in DCOP literature (Farinelli et al., 2008). In this type of problem, each agent controls one variable defined over a set of colours. The variables are connected within a graph and the aim of the problem is to minimise the number of connected variables assigned to the same colours (i.e. the number of conflicts). The key idea of this benchmark is that the agents need to coordinate and change the assignment of their variables (i.e. their decision) to minimise the number of conflicts. Thus, given the number of agents and topologies, a variety of configurations exists. For this reason, the graph colouring problem reproduces many of the different conditions that define a coordination problem in the real world and thus, it constitutes a well acknowledged benchmark for DCOP algorithms.

Hence, we present here an extention of this benchmark to use it to represent multi-objective coordination problems. More formally, each agent $\mathcal{A}_j$ owns a single variable x_j, taking values in the domain $D_j = \{Red, Green, Blue\}$. Within this setting, the agents' goal is to maximise the following multi-objective function:

$$C(X) = \left[C^1(X), C^2(X), C^3(X)\right]^T \tag{5.10}$$

where C^1, C^2, C^3 are the sum of bi-variate constraint functions $C_j^1(x_i, x_k)$, $C_j^2(x_i, x_k)$, $C_j^3(x_i, x_k)$ that exist among the variables X. These three types of constraint functions are defined as follows:

Chromatic Difference: This objective function represents the common graph colouring conflict function:

$$C_j^1(x_i, x_k) = \begin{cases} 0 & x_i \neq x_k \\ -1 & x_i = x_k \end{cases} \tag{5.11}$$

Chromatic Ordering: This objective function imposes an ordering among the colours: $Red = 1$, $Green = 2$, and $Blue = 3$. Specifically, given two variables x_i and x_k where $i < k$, the variable with the higher index should have a higher ranked colour:

$$C_j^2(x_i, x_k) = \begin{cases} 0 & \text{if } i < k \text{ and } x_i < x_k \\ -1 & \text{otherwise} \end{cases} \tag{5.12}$$

Chromatic Distance: This objective is similar to the chromatic ordering. However, it considers the distance between the colours of different variables. In more detail, given two variables x_i and x_k, the distance of the colours between the two variables should equal one:

$$C_i^3(x_i, x_k) = \begin{cases} 0 & \text{if } |x_i - x_k| = 1 \\ -1 & \text{otherwise} \end{cases} \tag{5.13}$$

This three objectives present different conflicts. The chromatic difference and the chromatic ordering conflict because even if two variables are assigned different colours (i.e. the chromatic difference objective is satisfied) that does not necessarily mean that these colours are ordered correctly (i.e. that the chromatic ordering objective is satisfied). The same applies for the chromatic difference and the chromatic distance objectives: if the former is satisfied, that does not necessarily mean that the two variables are assigned to colours satisfying the latter objective (i.e. that they are assigned colours with a distance equal to 1). The chromatic ordering and distance objectives then conflict because if two variables are assigned to ordered colours, this does not necessarily mean that these colours share a distance equal to 1 (i.e. that the chromatic distance objective is satisfied). Hence solving a multi-objective graph colouring problem requires to trade-off between these different objectives to find those variable assignments that minimise the number of conflict between the different colours.

Thus, given an arbitrary graph $G = (V, E)$, we construct a factor graph as follows. Each vertex is represented as a variable x. Furthermore, for each edge (v, v') the factor graph contains a three-objective constraint function $C_j(x_i, x_k) = [C_j^1(x_i, x_k), C_j^2(x_i, x_k), C_j^3(x_i, x_k)]^T$ where x_i and x_k are the variables corresponding to vertices v and v'.

5.3.2 Experimental Setup

We analyse the performance of our algorithm by executing it on several instances of the multi-objective graph colouring problem defined previously. Specifically, we randomly generate *connected* graphs $G = (V, E)$ with a varying number $M = |V|$ of vertices, and varying graph density $\delta \in [0, 1]$. This latter parameter defines the constrainedness of the problem by controlling the number of edges; when $\delta = 0$ the number of edges $|E| = M - 1$, and the resulting graph G is a tree. Conversely, when $\delta = 1$, G is a complete graph, and $|E| = \frac{1}{2}M(M-1)$.

We generated problem instances with $M = \{8, 10, 12, 14, 20, 40, 60, 80, 100\}$ and $\delta = \{0.0, 0.1, \ldots, 1.0\}$. By varying the density and the number of nodes of the graph, we are able to control the interactivity of the agents. In so doing, we are able to reproduce the different levels of connectivity that might characterise the agents when deployed in the real world. For instance, while tracking one or more targets, it might happen that one or more UAVs are tracking the same target. The number of these interactions will vary depending on the number of targets and UAVs. Hence, by controlling the density of the graph, we are able to reproduce all the different type of interactions that can arise in such problems. Moreover, for each combination of values for M and δ we ran our algorithm 120 times to achieve statistical significance. We measure its performance using the following four metrics:

1. The runtime (in milliseconds) required by the algorithm to compute the set of solutions. This metric allows us to measure the amount of computation required by the agents to solve a MO-DCOP.

2. The average approximation ratio of the solutions PO_X computed by the algorithm. More specifically, we defined this as the norm of the approximation ratio vector $||\boldsymbol{\rho}||$ defined in Section 5.2.2. This metric allows us to measure quality of the solutions found by the algorithm.

3. The minimum Euclidean distance between solutions $X' \in PO_X$ computed by the algorithm and an optimal solution $X^* \in PO_X$. More formally, we calculate this distance as follows:

$$d(X') = \left\| \min_{X^* \in PO_X} \left[C(X') - C(X^*) \right] \right\|$$

   This metric allows us to quantify how far the solutions recovered by B-MOMS are from the Pareto optimal ones.

4. Finally, we measure the fraction of Pareto optimal solutions found by the algorithm:

$$n_{PO} = \frac{|\widetilde{\mathbf{PO}}_{\mathbf{X}} \cap PO_X|}{|PO_X|}$$

This metric tests how many of the recovered solutions are actually optimal. In so doing, it verifies the capability of B-MOMS to retrieve optimal solutions.

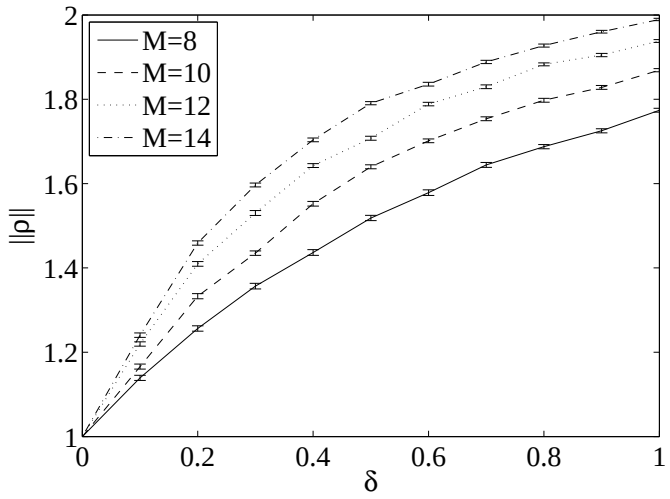
Note that metrics 2 and 3 aggregate the objectives by using the notions of norm and Euclidean distance, which implies commensurability of the objectives. However, these metrics have been widely used in multi-objective literature (Veldhuizen and Lamont, 1998). Moreover, metrics 3 and 4 require the availability of the set of Pareto optimal solutions PO_X. To compute these, we used a centralised canonical brute-force optimal algorithm. This optimal algorithm exhaustively enumerates the Cartesian product of the domains of $\mathbf{x}$, and thus, has a computational complexity that is exponential in the number of variables. As a result, we do not report metrics 3 and 4 for $M > 14$, since the algorithm was not capable to scale to more than 14 variables. However, we do report metrics 1 and 2 for M up to 100.

5.3.3 Results

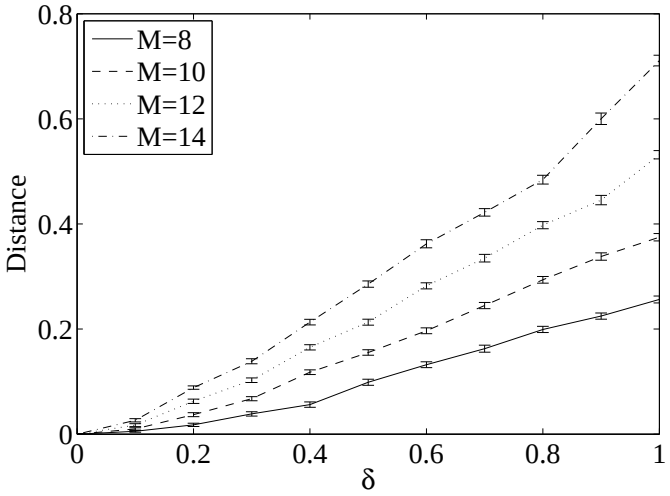
For $M \leq 14$, the results are shown in Figure 5.3. First of all, all three plots confirm that the algorithm is optimal for acyclic graphs ($\delta = 0$), as proven by Theorem 5.3. Considering then that Theorem 5.3 is a specialisation of Theorem 4.3, these results also confirm that algorithms exploiting the PO-GDL are optimal over acyclic factor graphs.

In addition, by increasing the number of constraints of the problem (i.e. by increasing δ), we can observe that the performance of B-MOMS degrades gracefully in terms of the approximation ratio, distance, as well as the fraction of optimal solutions found. Moreover, the approximation ratio never exceeds 2 (i.e. the value of the computed solution is greater than half of that of the optimal solution) even for extremely constrained problems, and is close to the value of 1.27 reported for the single-objective graph colouring problem by the single-objective bounded max-sum algorithm (Rogers et al., 2011) when each variable is involved in three constraints (corresponding to $\delta = 0.3$ for $M = 14$). Most importantly, for relatively sparse graphs ($\delta \approx 0.2$), which are often found in real-life multi-agent applications (Modi et al., 2005; Rogers et al., 2011), B-MOMS recovers roughly 50% of the optimal solutions for $M = 14$. This is remarkable, given the fact that the algorithm is approximate, and was not specifically designed to recover these. Intuitively, this happens because the spanning tree computation for sparse graphs is capable of maintaining some of the solutions of the original problem. Hence, the former three results give us an idea of the level of performance to expect when using the PO-GDL to instantiate a bounded approximate algorithm.

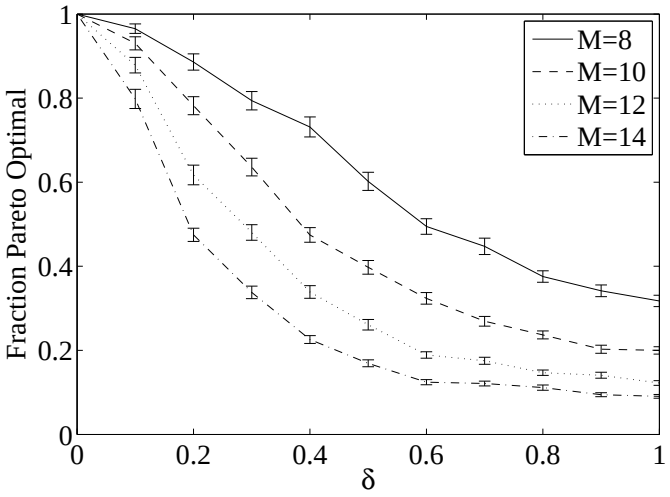
Such level of performance is further demonstrated in Figures 5.4(a) and 5.4(b) which report the approximation ratio and the runtime for larger problem instances. Specifically, Figure 5.4(a) clearly shows that, even for large instances, the approximation ratio again



(a) Approximation ratio $||\boldsymbol{\rho}||$



(b) Minimum distance of computed solutions to a Pareto optimal solution



(c) Fraction n_{PO} of Pareto optimal solutions found

FIGURE 5.3: Empirical results for $M \leq 14$. Errorbars indicate the standard error of the mean.

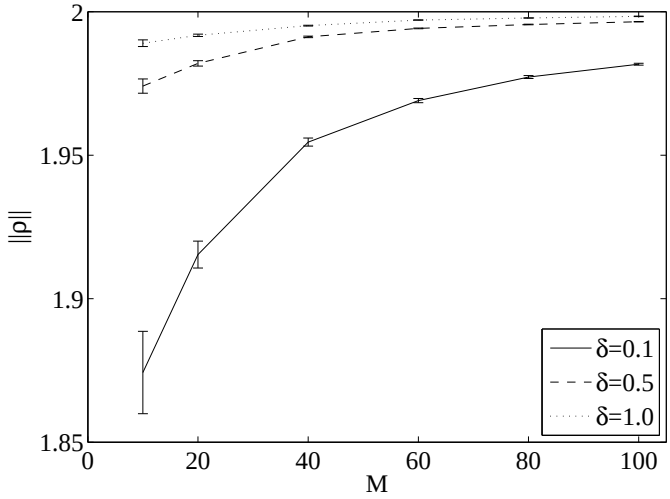
never exceeds 2, demonstrating the effectiveness of the bounding approach. Indeed, this value of the approximation ratio indicates that the values of the bounds, computed over each objective, never exceed half of the value of the corresponding optimal solution in the utopia point. Furthermore, Figure 5.4(b) gives strong empirical evidence of the practical applicability of the algorithm. Despite the exponential relation between the number of variables and the time required by B-MOMS[4], for $M = 100$ and a maximally constrained problem, this time does not exceed 30 minutes. Moreover, it is important to note that these experiments were run on a single processor, while the computational load in a multi-agent system is typically shared among multiple computational entities. This brings PO-GDL algorithms such as B-MOMS well within the realm of the limited computational capacities of embedded agents found in many real-life applications.

5.4 Summary

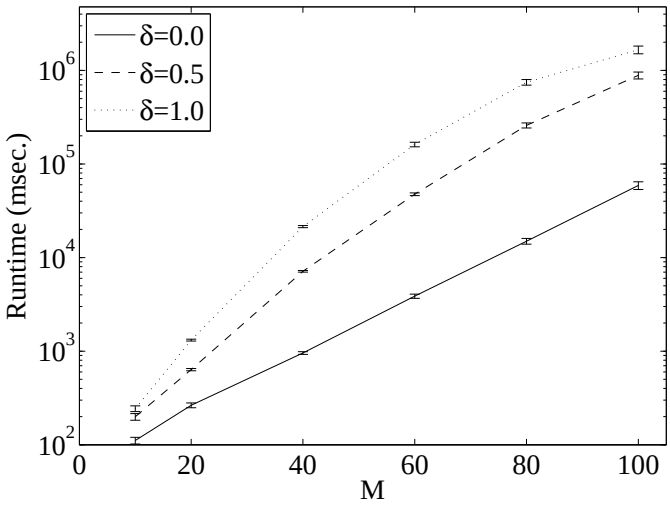
This chapter presented a new class of problems and solution techniques to address coordination problems that involve multiple objectives. More specifically, in Section 5.1, we proposed the MO-DCOP framework, a specialisation of the PO-DCOP framework defined in Chapter 4. The key idea of a MO-DCOP is to re-define the partially ordered constraint functions of a PO-DCOP into multi-objective functions. In so doing, we propose a class of problems that meet the complex interaction requirement defined in Chapter 1. In addition, the MO-DCOP framework also satisfies the generality requirement, since it can represent any type of problem where the agents need to coordinate over multiple objectives.

We then presented, in Section 5.2, a new set of solution techniques for solving MO-DCOPs. To derive these techniques, we specialised the PO-GDL and derived a new commutative semi-ring based on the notion of Pareto dominance (Chapter 2) which allowed us to instantiate the message passing algorithms presented in Chapter 2 (Algorithms 2.1 and 2.2) to address problems involving multiple objectives. The key idea of these new algorithms is that by using the concept of Pareto dominance they become capable of trading off between the different objectives of a MO-DCOP. In addition, the PO-GDL framework allows us to instantiate optimal or bounded approximate algorithms. In so doing, our techniques inherit the qualities of GDL algorithms discussed in Chapter 2 which are very useful in our work. In particular they meet the requirements of quality guarantees and of robustness since they are fully decentralised as discussed in Chapter 1.

[4]This is partially due to our implementation of the value-propagation phase, which for the purpose of these experiments, recovers an exponential number of (approximately) Pareto optimal solutions, instead of just a single one.



(a) The approximation ratio $||\boldsymbol{\rho}||$ for varying number of vertices



(b) The runtime for varying number of vertices

FIGURE 5.4: Empirical results for $10 \leq M \leq 100$. Errorbars indicate the standard error of the mean.

Next, to verify the way in which our algorithms satisfy the requirements of scalability and resource awareness, we presented in Section 5.3 an empirical evaluation of the B-MOMS algorithm which uses the previously defined semi-ring to generalise the bounded max-sum algorithm (see Chapter 2) for solving MO-DCOPs. The focus of our experiments was on verifying some of the fundamental properties of B-MOMS (and of our techniques in general) such as the optimality over acyclic factor graphs, the approximation ratio and the cpu time necessary to compute a set of solutions. To achieve this, we evaluated our B-MOMS over different problem instances of a multi-objective extension of the graph colouring problem, a well known benchmark for DCOP algorithms. Our results showed that our algorithm is capable of providing good approximate solutions in a reasonable runtime, even for extremely large and complex problems. This, in turn, indicates that the algorithm satisfies the requirements of scalability and resource awareness.

In the next chapter, we pursue our study of algorithms for solving coordination problems involving complex interactions and focus on *optimally* solving problems involving uncertainty and risk awareness.

Chapter 6

A Class of Algorithms for Partially Ordered Coordination Problems involving Risk Awareness

This chapter presents a formal framework to solve distributed constraint optimisation problems (DCOPs) in which the outcome of the constraint functions is uncertain. Considering then, that these functions are typically used to measure the quality of the collective decisions of a set of agents, the key aim of this chapter is to study problems in which the outcome of the agents' decisions is uncertain. In more detail, in this setting, the agents do not know with precision what the outcome of their decision is going to be. Hence, if they do not carefully weight each of their decisions, they are likely to make decisions which might result in a very poor outcome. Consequently, the agents need to take into account the collective risk (i.e. the potential that one possible collective decision can lead to an undesirable outcome) associated with the uncertainty of each possible decision.

In fact, uncertainty is endemic within many real world coordination problems. To understand this, we consider again the domain of disaster response, in which robotic agents are deployed to achieve situational awareness at the scene of a disaster (see Chapter 2). For instance they could be deployed to explore the scene of the disaster in search for casualties, a setting similar to the target search problem described in Chapter 2. In this setting, agents make decisions about which areas to explore and the key objective is to select the areas most likely to contain undiscovered casualties. However, many aspects of the agents and of the environment where they are deployed are uncertain. For instance, the agents' sensors may be noisy and their global positioning system (GPS)

may be imprecise. Thus, the agents cannot be certain about the number of casualties reported by their sensors. Moreover, the casualties' position is uncertain and for this reason this number can only be estimated using its expected value. In light of this, the outcome of the agents' decisions is uncertain since they do not know with precision the number of casualties contained within each area. Hence, it is advisable that these agents incorporate this uncertainty in their decision making and become capable of weighting their joint decisions to avoid poor outcomes (i.e. explore areas with a low number of casualties).

To improve the agents' capabilities and make them capable of effectively facing uncertain decisions a risk profile is typically used. Such a profile is defined as a utility function which typically encodes three types of attitude towards the risk of making decisions resulting in poor outcomes (see Chapter 1). A risk averse profile defines agents more inclined to make certain but less rewarding decisions, rather than highly rewarding (in expectation) but uncertain ones. Hence, by using this risk profile, the agents would prefer to explore an area containing a certain but lower number of casualties than an area containing an uncertain but higher number. In contrast, a risk seeking profile defines agents which prefer uncertain but more rewarding (in expectation) decisions over certain ones. In this case, the agents would prefer to explore areas with a large, but uncertain, number of casualties instead of areas with a certain but lower number. Finally, a risk neutral profile defines agents as being indifferent between certain or uncertain decisions. Thus, by adopting this type of profile, the agents will be indifferent to the uncertainty related in the number of casualties related to the areas that they are exploring.

Now, DCOPs extensions that model uncertain constraint functions have been proposed by literature (see Section 2.4.1). However, these frameworks still do not allow the agents' collective attitude towards risk to be modelled. Moreover, existing algorithms are usually approximate and do not provide any guarantee on the quality of the solutions recovered, all of which are key requirements for our work (Chapter 1).

To address these shortcomings, this chapter develops two main contributions. First, we instantiate PO-DCOPs to incorporate uncertainty and the agents' collective attitude towards risk. In so doing, we propose a novel general framework to allow agents to coordinate under uncertainty. We call this framework risk aware distributed constraint optimisation problems (RA-DCOPs). RA-DCOPs specialise partially ordered distributed constraint optimisation problems (PO-DCOPs) to the setting in which the agents' decisions are uncertain, and thus where the concept of risk need to be incorporated in the problem. Second, to solve RA-DCOPs, this chapter follows the same methodology used in Chapter 5 and described in Chapter 4. In essence it builds upon the partially ordered generalised distributive law (PO-GDL) framework to derive a class of algorithms that can solve RA-DCOPs. More specifically, we define the commutative semi-ring characterising the objective function of an RA-DCOP, the non-dominated /

convolution (NDC) semi-ring, by using the agents' risk profile to factorise the global constraint function and use it to instantiate a GDL algorithm (Action GDL) to solve RA-DCOPs. However, to solve these problems optimally, these algorithms have to keep track of multiple non-dominated partial solutions. Hence, we show that different types of domination conditions exist. Optimal conditions propagate the smallest number of probability density functions (pdfs) without losing the guarantee of optimality, but do not always have a closed form expression. Alternatively, sufficient (which achieve optimality at the cost of a higher overhead), and necessary (which incur lower costs but can sacrifice optimality) conditions can be used. Next, we empirically demonstrate the need for this new class of algorithms to solve RA-DCOPs by showing that transforming them into DCOPs can lead to significantly suboptimal performance, especially as the level of risk aversion is increased.

The remainder of this chapter proceeds as follows. Section 6.1 describes the general framework that we defined. Section 6.2 then presents the GDL-based algorithm which we derived to solve RA-DCOPs. Next, Section 6.3 presents some empirical evaluation and Section 6.4 summarises the chapter.

6.1 Problem Definition

This section describes the framework of RA-DCOPs, an instantiation of PO-DCOPs that incorporate the notion of collective risk awareness to the canonical DCOP framework. Formally, an RA-DCOP is defined as follows:

Definition 6.1. A RA-DCOP is a tuple $\langle A, X, D, F, C, U \rangle$ such that:

- $A = \{1, \ldots, |A|\}$ is a set of agents.
- $X = \{x_1, x_2, \ldots, x_n\}$ is a set of variables.
- $D = \{D_1, \ldots, D_n\}$ is a set of discrete domains, where D_i is the domain of variable x_i.
- $F : X \to A$ is a function that assigns variables to agents. Each agent controls (the value of) the variables that are assigned to it.
- $C = \{C_1, \ldots, C_m\}$ is a now set of *non-deterministic* (instead of deterministic scalar) constraint functions. Hence, the value of an assignment of X_j returned by constraint function C_j is a random variable V_j whose pdf is *known*, i.e. $V_j \sim C_j(X_j)$.
- $U : \mathbb{R} \to \mathbb{R}$ is the utility function of the entire collective of agents A encoding their collective attitude towards risk.

Thus, in a RA-DCOP, as is the case for a general PO-DCOP, the sets A, X, D and F maintain the same semantics as for traditional DCOPs. The constraint functions C_j are then generalised into non-deterministic functions. Hence, each assignment $C_j(X_j)$ is now a random variable V_j with a known pdf. This means that the different assignments $C_j(X_j)$ are partially ordered. As a consequence, there is no more a single optimal solution X^* as for canonical DCOPs. Rather, as discussed in Chapter 4, we have a PO-DCOP in which multiple non comparable or non-dominated solutions exist. In addition, the sum of the local constraint functions associated with an assignment to variables X is also a random variable $V = \sum_{j=1}^{m} V_j$, whose probability distribution is the convolution of the distributions of the outcomes V_j of the local constraint functions. Consequently, each global assignment to X results in a (possibly) different distribution of V. The solution to a RA-DCOP is an assignment X^* to variables X that maximises expected global utility of the sum of the local constraints:

$$X^* = \arg\max_X \mathbb{E}\left[U\left(\sum_{j=1}^{m} V_j\right)\right] \tag{6.1}$$

To understand the intuition behind a RA-DCOP, note that function U models the attitude of the entire multi-agent system (not individual agents) towards the risk associated with each possible variable assignment to X. To understand this, consider again the domain of disaster response discussed in Chapter 2. Within this setting, the agents' risk profile is encoded within a utility which clearly increases with the number of civilians saved, but first responders might prefer a smaller (in expectation) but highly certain number of saved civilians over a better (in expectation) but highly uncertain (thus risky) outcome. In general, it can be assumed that this U is non-decreasing, i.e. more utility is always preferred over less (Levy, 2006).

In more detail, when the system is risk neutral, U is linear, and the system maximises the sum of expected utilities of the local constraints. However, when this is not the case, for instance when the collective agents is either risk seeking or averse, Equation 6.1 cannot be expressed as a sum of factors and thus, in general:

$$\mathbb{E}\left[U\left(\sum_{j=1}^{m} V_j\right)\right] \neq \sum_{i=j}^{m} \mathbb{E}[U(V_j)] \tag{6.2}$$

Hence, a RA-DCOP cannot be transformed into a DCOP by simply defining $C_j(X_j) = \mathbb{E}[U(V_j)]$ (as is done by Atlas and Decker (2010)). Example 6.1 illustrates this:

Example 6.1. *Consider a RA-DCOP with* $A = \{A_1, A_2\}$, $X = \langle\{x_1, x_2\}$, $D = \{\{0,1\},\{0,1\}\}$, *and* $C = \{C_1(x_1, x_2), C_2(x_1, x_2)\}$. *Furthermore, let* $U(V)$ *be a function such that* $\mathbb{E}[U(V)] = \mu_V - \sigma_V$, *where* μ_V *and* σ_V^2 *are the mean and standard deviation of* V.

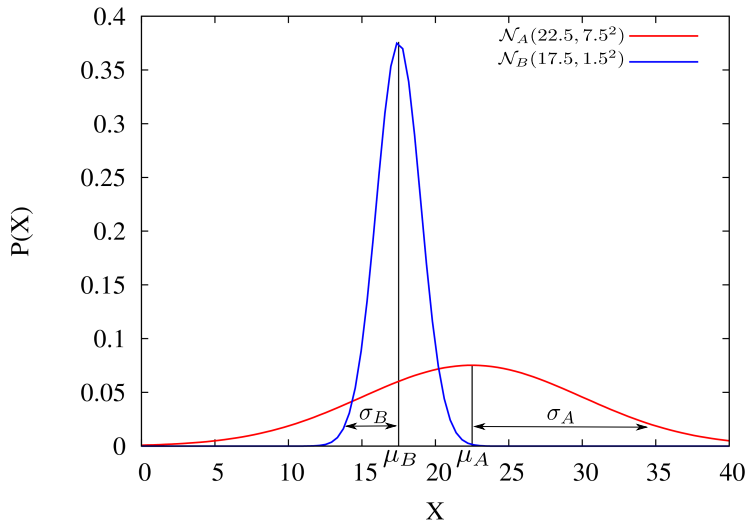


FIGURE 6.1: Two Gaussians corresponding to random variables A and B of Example 6.1

In the remainder of this work, we will use the utility function $U(V)$ to illustrate and empirically evaluate our algorithms. $U(V)$ is a well known type of risk averse utility function (see (Levy, 2006)) because it favours certain but less rewarding outcomes over more rewarding but highly uncertain ones. To understand this, consider two random variables $A \sim \mathcal{N}(22.5, 7.5^2)$ and $B \sim \mathcal{N}(17.5, 1.5^2)$. Their pdfs are shown in Figure 6.1. In this setting, if an agent selected a variable considering only its expectation, than A would be the better option since $\mu_A > \mu_B$. However, A has a larger deviation, therefore the outcome of the variable is more uncertain than B. In light of this, variable B is the best option since despite its lower expectation, it has a smaller deviation than A. Hence, the outcome of the variable B is more certain than that of variable A. This is reflected in the utility function since $\mathbb{E}[U(B)] = 16 > \mathbb{E}[U(A)] = 15$.

Constraint functions C are shown in Table 6.1. The last two columns represent the left-hand and right-hand side of Equation 6.2, i.e. the expectation of the utility of the sum of local constraints (EUS) and the sum of the expected utilities of the local constraints (SEU).

As indicated in the table, the solution to this RA-DCOP is $x_1 = x_2 = 0$ (see next to last column). Note that maximising SEU, rather than EUS, results in a suboptimal expected utility (-3 instead of 2). Thus, transforming a RA-DCOP into a DCOP using Equation 6.2 results in sub-optimality.

As suggested by Example 6.1, RA-DCOPs and DCOPs are two different classes of problems. In fact RA-DCOPs subsumes DCOPs and the following property holds:

Property. A DCOP is a RA-DCOP in which the outcome of each constraint function $C_j \in C$ is known with precision and where the risk function U is the identity function.

x_1	x_2	C_1		C_2		$C_1 + C_2$		EUS	SEU
		μ	σ^2	μ	σ^2	μ	σ^2		
0	0	9	8^2	10	15^2	19	17^2	**2**	-4
0	1	3	5^2	10	12^2	13	13^2	0	-4
1	0	15	7^2	5	24^2	20	25^2	-5	-11
1	1	2	4^2	2	3^2	4	5^2	-1	$\mathbf{-3}$

TABLE 6.1: The function payoffs of Example 2.2.

In light of this, we can define the commutative semi-ring $\langle R, \max_{\succ}, \mathbf{+} \rangle$, that characterise the objective function of an RA-DCOP (Equation 6.1) as was done in Chapter 5 for the case of multi-objective DCOPs. In so doing, we are essentially instantiating the operators $\max_{\succ}$ and $\mathbf{+}$ which we are going to use in Section 6.2 to define an algorithm for solving RA-DCOPs based on the PO-GDL.

Now, as discussed in Chapter 2, for any GDL-based algorithm to work the $\max_{\succ}$ operator needs to distribute over the $\mathbf{+}$ operator (Definition 2.2). However, in contrast with Chapter 5 where the notion of Pareto dominance was used, here we need to explicitly define $\max_{\succ}$. Hence, we impose a partial order over the set of random variables as follows:

Definition 6.2. NDC is a commutative semi-ring $\langle R, \max_{\succ}, \mathbf{+} \rangle$, such that:

- $R = \mathcal{P}(S)$ is the powerset (i.e. the set of all subsets) of S. Each set $S_i \in S$ contains all random variables, distributed according to a valid pdf.[1]
- $\max_{\succ}$ takes sets of random variables $S_1, S_2, \ldots, S_m \subseteq R$ and outputs a set S' such that:

$$\begin{aligned} S' &= \max_{\succ}\{S_1, S_2, \ldots, S_m\} \\ &= \left\{ V \in \bigcup_{i=1}^{m} S_i \mid \nexists V' \in \bigcup_{i=1}^{m} S_i : V' \succ V \right\} \end{aligned} \tag{6.3}$$

  where to ensure $\max_{\succ}$ distributes over $\mathbf{+}$, binary operator $\succ$ compares the expected utility of random variables X and Y under convolution with an unknown Z:

$$\forall X, Y \in S : X \succ Y \text{ iff} \quad \forall Z \in S,\ \mathrm{E}[U(X+Z)] \geq \mathrm{E}[U(Y+Z)] \tag{6.4}$$

  with strict inequality for at least one Z.

[1] The identity elements $\mathbf{0}$ and $\mathbf{1}$ are singleton sets with a random variable with an expected value of $-\infty$ and a random variable with a Dirac delta pdf, respectively. In a practical implementation, however, these elements can be replaced by special constants.

- $\mathbf{+}$ is a binary operator that, given two *sets* of random variables $S_1, S_2 \in R$, sums *all* pairs of random variables $V_1 \in S_1$ and $V_2 \in S_2$:

$$S_1 \mathbf{+} S_2 = \{V_1 + V_2 \mid \forall V_1 \in S_1, \forall V_2 \in S_2\} \tag{6.5}$$

  The sum of two random variables $V_1 + V_2$ is calculated using the convolution of the variables' corresponding pdfs. Hence, if V_1 and V_2 are distributed according to pdfs f and g, then the pdf of $V_1 + V_2$ is the convolution of f and g:

$$(f * g)(x) = \int_{-\infty}^{\infty} f(x-a)\dot{g}(a)da \tag{6.6}$$

To understand the intuition behind the $\max_{\succ}$ operator, we stated that it needs to distribute over the $\mathbf{+}$ operator, as per the definition of a commutative semi-ring (see Chapter 2). This implies that the ordering operator $\succ$ should also distribute over the convolution of random variables. However, achieve this is not trivial. For instance, if we simply define $X \succ Y =$ true iff $E[U(X)] \geq E[U(Y)]$, then $\max_{\succ}$ does not distribute over $\mathbf{+}$ anymore. To understand this, consider three variables X, Y and Z, we have that $X + Z \succ Y + Z$ iff $E[U(X+Z)] > E[U(Y+Z)]$. However, $E[U(X)] + E[U(Z)] > E[U(Y)] + E[U(Z)] \not\Leftrightarrow E[U(X+Z)] > E[U(Y+Z)]$, unless U is linear. Thus $\succ$ does not distribute over the convolution and so does $\max_{\succ}$ over $\mathbf{+}$. This means that Equation 6.4 needs to be defined consistently with the specification of a semi-ring. Moreover, the $\succ$ operator imposes a partial order as described by the following example:

Example 6.2. *Let $U(x)$ be the utility function from Example 6.1, and let the set S be restricted to normally distributed variables. The use of this assumption allows us to exploit the fact that the normal distribution is closed to convolution. More formally, for any two random variables $X \sim \mathcal{N}(\mu_X, \sigma_X^2)$ and $Y \sim \mathcal{N}(\mu_Y, \sigma_Y^2)$, their convolution is defined as $X + Y \sim \mathcal{N}(\mu_X + \mu_Y, \sigma_X^2 + \sigma_Y^2)$.*

In light of this, $X \succ Y$ iff $\mu_X + \mu_Z - \sqrt{\sigma_X^2 + \sigma_Z^2} \leq \mu_Y + \mu_Z - \sqrt{\sigma_Y^2 + \sigma_Z^2}$ for any $\mu_Z, \sigma_Z \in \mathbb{R}$. Algebraic manipulation shows that this inequality holds iff:

$$\mu_X - \mu_Y \leq \max(0, \sigma_X - \sigma_Y) \tag{6.7}$$

To see why $\succ$ is a partial order, consider $X \sim \mathcal{N}(0, 5^2)$ and $Y \sim \mathcal{N}(27, 35^2)$. Clearly, the inequality in Equation 6.7 does not hold, so $X \not\succ Y$ (or vice versa). To see why this is true, consider $Z_1 \sim \mathcal{N}(0, 0)$ and $Z_2 \sim \mathcal{N}(0, 12^2)$. Now note that $\mathrm{E}[U(X + Z_1)] = -5$ which is greater than $\mathrm{E}[U(Y + Z_1)] = -8$. However, $\mathrm{E}[U(X + Z_2)] = -13$ is less than $\mathrm{E}[U(Y + Z_2)] = -10$. Thus, discarding Y in favour of X (or vice versa) can result in sub-optimality.

To solve RA-DCOPs then, we adopt the same methodology described in Chapter 4. We define a commutative semi-ring so that we can factorise the objective function of an

RA-DCOP and we instantiated algorithms based on the PO-GDL. In what follows, we describe these algorithms.

6.2 An Algorithm for solving RA-DCOPs based on the PO-GDL

We discussed in Chapter 4, how the objective function of a PO-DCOP possesses a specific algebraic structure (i.e. a commutative semi ring) which can be used to derive various decentralised algorithms to solve it, all based on the GDL framework. In the same chapter, we also discussed the way the former algorithms can be instantiated by following the methodology used to derive canonical DCOP algorithms, such as DPOP, Action GDL and bounded max-sum (see Chapter 2).

In this chapter, we are interested in solving RA-DCOPs in which uncertainty and risk awareness need to be taken into account. To solve these problems, this section proposes a generalisation of Action GDL, a well known optimal algorithm to solve canonical DCOPs which has been shown to be more general than DPOP, the most well-known optimal GDL-based DCOP algorithm (Vinyals et al., 2011) (see Chapter 2). In so doing, we derive an optimal algorithm, which clearly incurs larger communication and computation costs. Nonetheless, as we will see in the remainder of this section, this complexity is required to study the properties of this new class of problems and approximate solutions will be considered for future work. In more detail, we propose an algorithm proceeding over three phases, as follows:

Phase 1 - Junction Tree Computation : The *junction tree* (JT) phase produces a junction tree of the factor graph encoding an RA-DCOP, using the same procedure of the canonical Action GDL algorithm.

Phase 2 - Message Passing : The *message passing* (MP) phase is a re-definition of the canonical GDL message passing phase to the setting in which the agents coordinate to find the non-dominated set of solutions to the cycle-free factor graph computed in the JT phase.

Phase 3 - Value Propagation : The *value-propagation* (VP) phase allows the agents to select a consistent variable assignment, considering that now multiple non-commensurable alternatives exist.

In what follows, we describe each of these phases in detail, and then we analyse some of the algorithm's key properties.

6.2.1 Algorithm Description

Our algorithm proceeds in three phases. In what follows, we discuss each of the phases in detail.

6.2.1.1 The Junction Tree Phase

In contrast, to the algorithm proposed in Chapter 5, no extensions to the first phase are required. Thus, we run the same junction tree algorithm as for the canonical Action GDL algorithm (Vinyals et al., 2011). In essence, the nodes of the original factor graph that represent variables are clustered together until the graph becomes acyclic. Each cluster of nodes is then replaced by a new node representing a meta-variable defined over the Cartesian product of the domains of the variables (see Chapter 2 for more details).

In what follows, we present an example that shows how the RA-DCOP described in Example 6.1 can be cast into a factor graph and transformed into a junction tree.

Example 6.3. *The RA-DCOP in Example 6.1 describes a setting with two variables x_1 and x_2, and two constraints C_1 and C_2. This can be encoded in the factor graph depicted in Figure 6.2(a).*

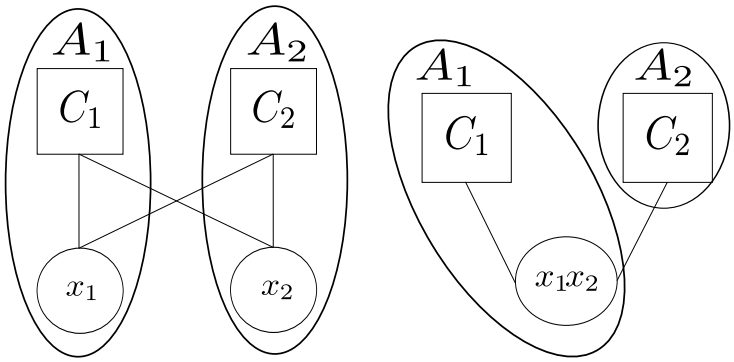
Now, to compute a junction tree, variables x_1 and x_2 are merged into a new variable x_p, the new RA-DCOP is defined as follows:

x_p	C_1		C_2		$C_1 + C_2$		*EUS*	*SEU*
	μ	σ^2	μ	σ^2	μ	σ^2		
$a = (0,0)$	9	8^2	10	15^2	19	17^2	**2**	-4
$b = (0,1)$	3	5^2	10	12^2	13	13^2	0	-4
$c = (1,0)$	15	7^2	5	24^2	20	25^2	-5	-11
$d = (1,1)$	2	4^2	2	3^2	4	5^2	-1	$\mathbf{-3}$

The resulting acyclic factor graph is shown in Figure 6.2(b) and constitutes the input of the second phase of the algorithm, which we discuss next.

6.2.1.2 The Message Passing Phase

The specification of the message passing phase proceeds as defined in Chapter 4 and as used in Chapter 5 for the case of multi-objective distributed constraint optimisation problems (MO-DCOP).



(a) The factor graph from Example 6.1

(b) The junction tree from the factor graph in Figure 6.2(a)

In terms of implementation, Definition 6.2 raises a number of practical issues. First, there exists no closed form expression for the convolution of arbitrary pdfs (Equation 6.5). To solve this, an approximate technique such as sampling or curve fitting may be used, or set S can be restricted to certain classes of distributions that are closed under convolution, such as Gaussian (as in Example 6.2), Gamma or Poisson distributions. In disaster response, Gaussian are typically used to represent the multiple sources of uncertainty, since typically, only the means and the variances of the pdfs of these sources are known (Thrun et al., 2006), and as a consequences the former Gaussians have the largest entropy and thus, are the best pdf to use following the axiom of maximum entropy (Thrun et al., 2006).

Example 6.4. *In order to keep the exposition as clear as possible, we consider a simple message passing schedule, where the computation starts at the leaf nodes and goes up through the different nodes of the graph. The computation begins at time step $t = 0$, nodes C_1 and C_2 both have a single edge (i.e. they are leaves of the cycle free factor graph) and therefore send a message first:*

$R_{1\to p}(x_p) =$

x_p	C
$a = (0,0)$	$[9,8]$
$b = (0,1)$	$[3,5]$
$c = (1,0)$	$[15,7]$
$d = (1,1)$	$[2,4]$

$R_{2\to p}(x_p) =$

x_p	C
$a = (0,0)$	$[10,15]$
$b = (0,1)$	$[10,12]$
$c = (1,0)$	$[5,24]$
$d = (1,1)$	$[2,3]$

Both C_1 and C_2 compute the multi-objective marginal function of variable x_p, as detailed above. At time step $t = 1$, x_p receives the messages and computes the next two messages:

$$Q_{p\to 1}(x_p) = \begin{array}{|c|c|}\hline x_p & C \\ \hline a=(0,0) & [10,15] \\ b=(0,1) & [10,12] \\ c=(1,0) & [5,24] \\ d=(1,1) & [2,3] \\ \hline \end{array} \qquad Q_{p\to 2}(x_p) = \begin{array}{|c|c|}\hline x_p & C \\ \hline a=(0,0) & [9,8] \\ b=(0,1) & [3,5] \\ c=(1,0) & [15,7] \\ d=(1,1) & [2,4] \\ \hline \end{array}$$

After these messages have been sent, the MS phase terminates. Finally, variable x_p can compute its non-dominated values:

$$ND_X = \max_{\succ} \left\{ \begin{array}{|c|c|}\hline x_p & C \\ \hline a=(0,0) & [19,17] \\ b=(0,1) & [15,13] \\ c=(1,0) & [20,25] \\ d=(1,1) & [4,5] \\ \hline \end{array} \right\}$$
$$= \{[19,17]\}$$

Such values correspond to the non dominated solutions of the problem $O_X = \{x_p = (0,0) = a\}$.

Similar to canonical DCOP algorithms, it can happen that different solutions yield the same value of the global constraint function. This is even more true for MO-DCOP where there might exist numerous non-comparable solutions which are all Pareto optimal. For this reason, a value propagation phase is required, which we discuss next.

6.2.1.3 The Value-Propagation Phase

The value-propagation phase operates on the cycle-free factor graph computed in the first phase. Variables and function nodes in the factor graph coordinate to select a consistent variable assignment. Since, RA-DCOPs are PO-DCOPs then, we can use the value propagation described in Chapter 4, by instantiating the $\succ$ relation using the dominance condition defined in the previous section (or the approximation which we will discuss in Section 6.2.2).

However, in contrast to the example presented in Chapter 5, it should be noted that since the solution of Example 6.1 is unique, the value propagation phase is not necessary in this case.

6.2.2 Theoretical Analysis

We now discuss some of the fundamental properties of GDL-based algorithms that we can instantiate using the NDC semi-ring (Definition 6.2).

6.2.2.1 Optimality of the Message Passing Phase

In order to use the NDC-GDL to derive new algorithms, as described in Chapter 2, the following theorem must hold:

Theorem 6.3. *Given a RA-DCOP $\langle A, X, D, F, C, U\rangle$, if the problem's corresponding factor graph is acyclic and the stopping criterion is chosen such that Algorithms 2.3 and 2.4 are run for a number of iterations equal to the diameter of the factor graph, the following equation holds for each variable $x_i \in X$:*

$$Z_i(x_i) = \max_{\succ} \sum_{j=1}^{m} V_j \tag{6.8}$$

Proof. The proof follows from Theorem 4.3 and from the fact that the objective function of a MO-DCOP is defined over the NDC-GDL semi-ring, which is a commutative semi-ring. Thus, by instantiating the operator $\otimes$ of Algorithms 1 and 2, with the operator $+$ and by defining the dominance relation $\preceq$ depending on the type of risk profile considered (see Chapter 2), it follows that Equation 6.8 specialises Equation 4.6 to RA-DCOPs and thus, that the theorem holds. □

Here, operator $\max_{\succ}$ discards *dominated* partial solutions, i.e. those that are guaranteed to be suboptimal in terms of Equation 6.1. Operator $+$ ($\sum$ for more than 2 elements) sums random variables by computing the convolution of their pdfs. As a result, $Z_i(x_i)$ can contain *multiple* pdfs, each corresponding to a non-dominated global outcome $V \sum_{j=1}^{m} V_j$. The optimal assignment x_i^* of x_i can be recovered as follows:[2]

$$x_i^* = \arg\max_{x_i} \left[\max_{V \in Z_i(x_i)} \mathrm{E}[U(V)] \right] \tag{6.9}$$

Now, as stated in the theorem computing the operator $\succ$ depends on the type of risk profile used, and, consequently it might become difficult to derive a closed form solution. The next section discusses ways to address this shortcoming.

[2]Similar to standard DCOPs, this is only the case if the optimal solution is unique. Otherwise, value propagation should be used.

6.2.2.2 Sufficient and Necessary Conditions for Dominance

There exists no closed form conditions for the $\succ$ operator (Equation 6.4) for arbitrary combinations of utility functions U and random variables S. However, it is possible to derive alternative conditions. Indeed, the dominance condition in Equation 6.4 can be made weaker or stronger (i.e. discarding more or less pdfs), resulting in a sufficient and a necessary condition respectively.

Definition 6.4 (Sufficient Condition for Dominance)**.** A condition C that imposes relation $\succ_C$ is called *sufficient* if $X \succ_C Y \Rightarrow X \succ Y$.

A sufficient condition keeps all non-dominated random variables, but might not filter out all dominated random variables. Therefore, using a sufficient condition guarantees that the optimal (i.e. non-dominated) random variable is preserved, and consequently, that optimality is guaranteed. However, many dominated pdfs might be propagated in the messages exchanged between functions and variables, wasting computation and communication. To understand this with an example, consider the condition presented in Example 6.2 (Equation 6.7). An example of a sufficient condition is C:

$$\mu_X \geq \mu_Y \wedge \sigma_X \leq \sigma_Y \text{ (with strict inequality in at least one clause)} \tag{6.10}$$

To understand why this condition is weaker than Equation 6.7, consider two random variables $X \sim \mathcal{N}(0, 5^2)$ and $Y \sim \mathcal{N}(2, 4^2)$. By calculating Equation 6.7, we can see which variable (X or Y) is the best considering the risk utility function defined in Example 6.1. By doing so, we have that $X \succ Y$. However, by calculating Equation 6.10 for X and Y, we have that $X \not\succ_C Y$. Hence, using condition C, X and Y are equivalent and are both propagated despite $X \succ Y$.

Definition 6.5 (Necessary Condition for Dominance)**.** A condition C that imposes relation $\succ_C$ is called *necessary* if $X \succ Y \Rightarrow X \succ_C Y$.

A necessary condition discards all dominated random variables, but might discard non-dominated ones as well. Thus, a necessary condition can result in loss of optimality. Hence, using this condition, instead of a sufficient condition, implies that less pdfs (all non-dominated) are propagated in the messages exchanged between variables and functions. Thus, computation and communication are reduced. However, this also means that not all the non-dominated pdfs are propagated. To understand this with an example, consider again the condition presented in Example 6.2 (Equation 6.7). An example of a sufficient condition is D:

$$\mu_X - \sigma_X > \mu_Y - \sigma_Y \tag{6.11}$$

To understand why this condition is stronger than Equation 6.7, consider again two random variables $X \sim \mathcal{N}(0, 5^2)$ and $Y \sim \mathcal{N}(-2, 4^2)$. Following the same procedure as for the sufficient condition C, we have that $X \succ_D Y$, but not $X \succ Y$. In addition, it should be noted that condition D imposes a total order over the set of random variables. Hence, using this condition is equivalent to applying a DCOP algorithm to maximise the right-hand side of Equation 6.2, which was shown to be suboptimal (Example 2.2).

Definition 6.6 (Optimal Condition for Dominance)**.** A condition O that imposes relation $\succ_O$ is called *optimal* if it is both necessary and sufficient.

Equation 6.7 is an example of an optimal condition. To summarise, the key difference between a sufficient and a necessary condition is that the former is guaranteed to lead to optimal solutions, but also requires larger messages and more computation. Conversely, the latter does not guarantee optimality, but is computationally less expensive. When an optimal condition can be used, however, the smallest set of pdfs is propagated, without discarding any non-dominated random variables, as we show in the next section.

6.3 Empirical Evaluation

We study the properties of RA-DCOPs by using an instance of the NDC-GDL, as described in Section 6.2. The focus of these experiments is to study the properties of RA-DCOPs and the NDC-GDL class of algorithms across all possible probability distributions, and instantiations of the utility function U and NDC-GDL.

As discussed in the previous section, the behaviour of any NDC-based algorithm varies depending on whether the main, a sufficient or a necessary condition is used. Moreover, whereas any of these algorithms is optimal when instantiated with an optimal or sufficient condition, it is not possible to theoretically determine the impact on the execution time of the algorithm, or the communication requirements. Thus, we need to resort to empirical evaluation. Hence, the objective of this section is to empirically justify the need for NDC-GDL for solving RA-DCOPs (as suggested in Example 2.2), by quantifying the loss of solution quality that results from using standard DCOP algorithms. To do this, we compare our algorithm with the canonical Action-GDL algorithm, which maximises the right-hand side of Equation 6.2. Additionally, we compare the optimal solution of an RA-DCOP with the solution obtained by ignoring risk by setting $U(v) = v$ (which maximises $\mathrm{E}[V]$ instead of $\mathrm{E}[U(V)]$). By doing so, we can simultaneously ascertain the performance of an instance of the NDC-GDL class of algorithms, which explicitly takes the agents' risk profiles into account, against a standard GDL algorithm (Action-GDL), which does not.

In what follows, we discuss the empirical setup, the benchmarks and the empirical results.

6.3.1 Experimental Setup

We randomly generate RA-DCOPs with a varying number of three-valued variables. The reason for this is that the impact of the domain of the variables on the algorithm's performance is the same as all GDL-based algorithms. Thus it is irrelevant for the aims of this work. The number of functions m is controlled by density parameter $\delta \in [0, 1]$. For $\delta = 1$ there is a pairwise constraint function between all pairs of variables ($m = \binom{n}{2}$), while for $\delta = 0$, the factor graph is a tree.

By controlling the density parameter and the number of nodes of the graph, we follow the same procedure as in Chapter 5. The idea is to control the interactivity of the agents. In so doing, we are able to reproduce the different levels of connectivity that characterise the agents when deployed in the real world. For instance, while searching for one or more lost targets, it might happen that one or more UAVs are searching the same area. This means that their observations might overlap (i.e. they are making redundant decisions) as discussed in Chapter 2. Thus, the number of these interactions will vary depending on the number of overlapping observations and UAVs. Hence, by controlling the density of the graph, we are able to reproduce all the different type of interactions that can arise in such problems.

For each pair of variable assignments, the value assigned by a function is a normally distributed random variable with μ and σ^2 drawn from the intervals $[-1, 1]$ and $[0, \sigma^2_{\max}]$ with uniform probability. We varied σ^2_{max} between 1 and 10 to determine the effect of increasing levels of uncertainty (and risk) on the difference between the optimal solution of a RA-DCOP and its corresponding DCOP. The idea here is to create a controllable setting that can cover a range of different possible uncertain outcomes. In the context of robotics for disaster response, such outcomes could represent the uncertainty over the position of a target, over the number of casualties within an area or over temperature of a building (see Chapter 2 for more details).

We run two experiments. First, we use the utility function defined in Example 6.1 with the three dominance conditions defined by Equation 6.7, 6.10 and 6.11:

Optimal: $X \succ Y \Leftrightarrow \mu_X - \mu_Y \geq \max(0, \sigma_X - \sigma_Y)$.

Sufficient: $\mu_X \geq \mu_Y \wedge \sigma_X \leq \sigma_Y \Rightarrow X \succ Y$.

Necessary: $X \succ Y \Rightarrow \mu_X - \sigma_X \geq \mu_Y - \sigma_Y$. This is equivalent to using Action-GDL for maximising the right-hand side of Equation 6.2, which is a DCOP with $C_j(\mathbf{x}_j) = \mathrm{E}[U(V_j)]$. (Note that it is not the same as risk neutrality.)

The key idea here is to ascertain the communication and computational load used by our GDL-based algorithm when using a sufficient condition. Indeed, as we stated in the previous section, a sufficient condition is more likely to be used for solving RA-DCOPs since optimal conditions might not always be derived. However, the use of such a condition might come with an increment in terms of the computation and the communication required by the algorithm. Hence, we compare here a sufficient and an optimal condition to understand their differences. We also consider a necessary condition, which is the equivalent of instantiating an optimal DCOP algorithm, but which most likely will result in suboptimal solution quality as explained in Section 6.2.2.

Second, we use the utility function $\mu - \sigma^\gamma$ and compare the optimal solution of an RA-DCOP against the solution obtained when ignoring risk by assuming risk neutrality (i.e. $U(v) = v$). In these experiments, we vary σ in the interval $[1, 2]$, and γ between 0 (risk neutral) and 2 (risk averse). In so doing, the key idea is to study the impact of increasing risk averseness on the solution quality and required overhead of NDC-Action-GDL.

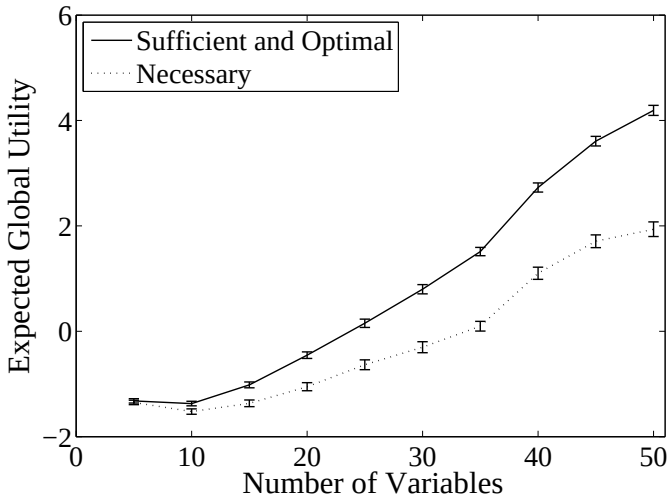
We randomly generate 200 problem instances to achieve statistical significance. In the first type of experiments, we measure the solution quality as per Equation 6.1, the average message size (in bytes), and the required computation time (in seconds). In the second, we report the error of ignoring risk by measuring the difference between using an optimal against necessary condition, as well as the savings in overhead in terms of computation and communication, again measuring the difference between the optimal and the necessary condition.

6.3.2 Results

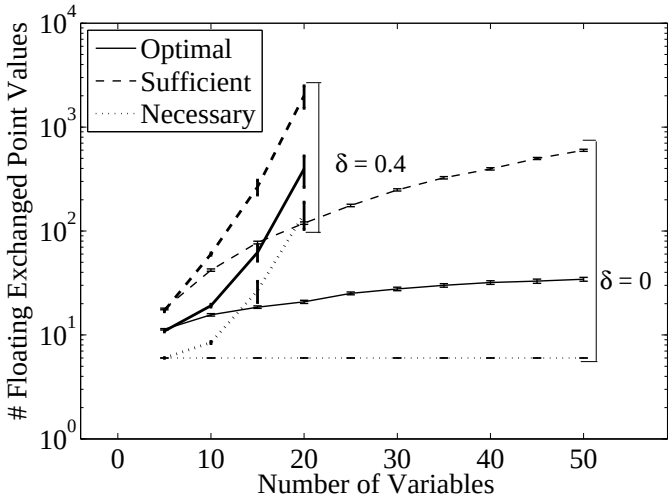
The results for the first experiment are shown in Figure 6.2.[3] These results clearly show the theoretical properties of the various dominance conditions: NDC-Action-GDL with a sufficient and an optimal condition is optimal, while Action-GDL (i.e. the necessary condition) is not (Figure 6.2(c)). Léauté et al. observed a similar result for their (incomplete) E-DPOP algorithm when using a linear decomposition (i.e. rhs of Equation 6.2) to maximise non-linear objective functions (Léauté and Faltings, 2011).

We found that the difference between the solution quality of using the optimal and the necessary condition increases as we increased $\sigma^2_{\max}$, with a mean absolute difference of 0.97 ($\pm$0.12) for $\sigma^2_{\max} = 1$ and 2.82 ($\pm$0.32) for $\sigma^2_{\max} = 10$ (for $n = 50$). Thus, as expected, Action-GDL algorithm performs worse as uncertainty increases. However, Figure 6.2(d) shows that this optimality comes at a cost of a roughly fivefold increase in message size for the optimal condition ($n = 40$), and a hundredfold increase for the sufficient condition ($n = 50$), as compared to Action-GDL. Simultaneously, the runtime

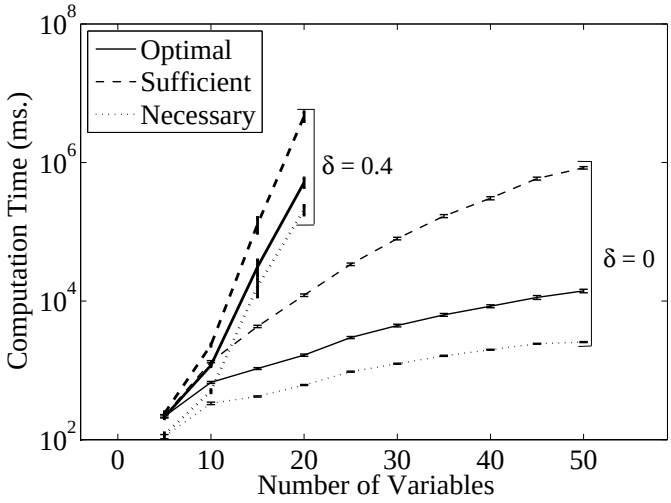
[3] We show results for $\delta = 0$ and $\delta = 0.4$ only, because for $\delta > 0.4$, the resulting junction tree clusters too many variables resulting in (NDC-)Action-GDL running out of memory.



(c) Solution quality



(d) Communication overhead



(e) Runtime

FIGURE 6.2: Empirical results. Error bars indicate the 95% confidence intervals.

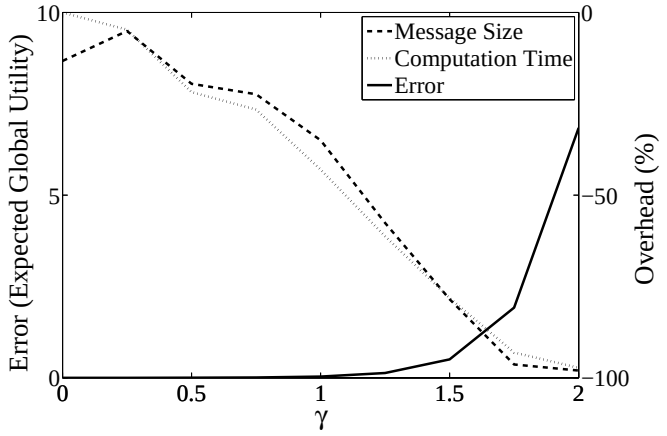


FIGURE 6.3: Error and reduction in overhead of ignoring risk.

of the optimal and sufficient conditions grows by a factor of 5 and 330 respectively (Figure 6.2(e)). However, depending on the application domain, a computational cost of less than 14 seconds (optimal) shared among 50 agents can be justified for a twofold increase in solution quality ($n = 50$).

The results of the second experiment are shown in Figure 6.3. The most evident fact here is that, as the agents become more risk averse (increasing γ), the error of ignoring risk increases. Notably, the error grows exponentially as $\gamma > 1$. The reason for this is that γ is the exponential factor in the utility $\mu - \sigma^{\gamma}$ and as soon as it becomes larger than one the risk averseness of the agents starts growing exponentially. However, the overhead in both computation and communication is significantly decreased by doing so. This means that as the difference between the performance of the optimal and the necessary conditions becomes larger, the overhead in terms of computation and communication becomes smaller. The reason for this is that the larger γ, the more the risk profile constrains the attitude of the agents towards uncertainty. Hence, a larger number of pdfs become non-dominated (i.e. the number of solutions grows larger), and thus, more alternatives need to be considered while using the optimal condition.

6.4 Summary

This chapter presented a new class of problems and solution techniques to address coordination problems involving uncertainty and risk awareness. More specifically, in Section 6.1, we proposed the RA-DCOP framework, a specialisation of the PO-DCOP framework from Chapter 4. The key idea of a RA-DCOP is that each outcome of the partially ordered constraint function of a PO-DCOP becomes a random variable defined by a specific pdf. Hence, in addition to multi-objective distributed constraint optimisation problems presented in Chapter 5, we propose here a further class of problems that meet

the complex interaction requirement defined in Chapter 1. In addition, the RA-DCOP framework also satisfies the generality requirement, since no conditions are put on the type uncertainty and risk profile.

We then presented, in Section 6.2, a new set of solution techniques for solving RA-DCOPs. To derive these techniques, we specialised the PO-GDL and derived a new commutative semi-ring based on the relation of non-dominance, which we need to be derived for specific pdfs and risk profiles. In so doing, we can then instantiate the message passing algorithms presented in Chapter 2 (Algorithms 2.1 and 2.2) to solve RA-DCOPs. By so doing, as described in Chapter 4, we are able to instantiate optimal or bounded approximate algorithm and to meet the requirements of quality guarantees and robustness defined in Chapter 1.

Now, to solve RA-DCOPs optimally, these algorithms have to keep track of multiple non-dominated partial solutions. To do so, in Section 6.2.2, we derived three types of domination conditions. Specifically, optimal conditions propagate the smallest number of probability density functions (pdfs) without losing the guarantee of optimality, but do not always have a closed form expression. Alternatively, sufficient (which achieve optimality at the cost of a higher overhead), and necessary (which incur lower costs but can sacrifice optimality) conditions may be used.

Finally, in Section 6.3, we empirically demonstrated the need for this new class of algorithms to solve RA-DCOPs by showing that transforming them into DCOPs can lead to significantly suboptimal performance, especially as the level of risk aversion is increased. However, this evaluation also shows that using our algorithms incurs a significant increase in terms of computation and communication. This is to be expected since we are solving optimally a problem which we have shown in Chapter 4 to be more complex than canonical DCOPs. Moreover, considering the performance loss derived from using canonical DCOP techniques to solve RA-DCOPs, such an increase in both computation and communication is to be expected. For this reason, despite being able to scale up to 50 agents (i.e. they meet the scalability requirement), our algorithm partially satisfies the requirement of resource awareness.

Chapter 7

A Class of Algorithms for Partially Ordered Coordination Problems involving Online Learning

This chapter presents a formal framework to solve distributed constraint optimisation problems (DCOPs) where the outcome of the constraint functions is uncertain (with unknown probability) and where no prior information is available about it. Considering then, that these functions are typically used to measure the quality of the collective decisions of a set of agents, the key aim of this chapter is to study problems in which the outcome of the agents' decisions is uncertain and unknown before-hand. In so doing, we aim to address the requirement of complex interactions described in Chapter 1 in the case where the agents need to learn the outcome of their decision online.

In more detail, in Chapter 2, we presented a number of algorithms to address uncertainty and lack of prior information. In the same chapter, we argued that, thus far existing literature lacks algorithms capable of addressing both of these challenges simultaneously. Moreover, existing algorithms do not provide any guarantee on the quality of the solutions recovered, which is a key requirement for the purpose of our work (Chapter 1). This is a clear limitation, since in many real world coordination problems not only is uncertainty endemic, but often little information is available about the environment where the agents are deployed. For instance, referring back to the scenario described in Chapter 1, the scenario describes the environment where the agents are deployed as unknown, uncertain and unpredictable. Moreover, these agents have noisy sensors. Thus, the outcome of their decisions cannot be defined *a priori* due to the level of uncertainty and the lack of knowledge. These outcomes then need to be learnt during the operation.

Against this background, this chapter proposes two main contributions. First, we first generalise the DCOP framework to the setting in which the values of the constraint functions is unknown. In more detail, we model each of these functions as a multi-arm bandit. This effectively models both the stochastic nature of realistic decentralised coordination problems, as well as the absence of *a priori* knowledge. To model these problems, we build upon the PO-DCOP framework presented in Chapter 4. More specifically, we specialise partially ordered distributed constraint optimisation problems (PO-DCOPs) to model problems in which the agents need to learn online the outcome of their decisions. In so doing, we enrich the canonical framework and allow it to be used for a broader class of real world problems. Second, to solve these problems, we propose a class of algorithms based on the generalised distributive law (GDL). Again, we build upon the framework described in Chapter 4 and instantiate the PO-GDL to derive algorithms to solve the previous problems. In so doing, we further generalise multiple canonical DCOP algorithms such as Action GDL, bounded max-sum and DPOP, whose level of performance has been well studied in literature, to be able to address decision making problems involving online learning. These algorithms solve MAB-DCOPs, an extension of canonical DCOPs, in which each local utility function becomes a multi-armed bandit (MAB) (see Chapter 2). Unlike a DCOP, a MAB-DCOP is not a single shot optimisation problem, but rather a sequential problem in which agents need to coordinate their joint actions over multiple time steps, so as to maximise the cumulative global utility received over a finite time horizon. Our algorithms achieve this by repeatedly choosing the joint action with the highest estimated upper confidence bound (UCB) on the sum of local utilities received in a single time step, which is a non-linear combination of the confidence bounds on the local utilities.

The remainder of this chapter proceeds as follows. Section 7.1 details the general framework that we defined. Section 7.2 then presents the GDL-based algorithm which we derived to solve MAB-DCOPs. Next, Section 7.3 presents some empirical evaluation and Section 7.4 summarises the chapter.

7.1 Problem Definition

This section describes the framework of multi-arm bandit DCOPs (MAB-DCOPs), a specialisation of PO-DCOPs that incorporate the concepts of UCB and MAB to the canonical DCOP framework. In so doing, this framework can be used to model problems in which the agents have to learn online the outcome of each of their decisions. Moreover, in so doing, MAB-DCOPs extend canonical DCOPs and address the requirement of complex interactions described in Chapter 1. Formally, a MAB-DCOP is defined as follows:

Definition 7.1. A MAB-DCOP is a tuple $\langle A, X, D, F, C \rangle$ such that:

- $A = \{1, \ldots, |A|\}$ is a set of agents.
- $X = \{x_1, x_2, \ldots, x_n\}$ is a set of variables.
- $D = \{D_1, \ldots, D_n\}$ is a set of discrete domains, where D_i is the domain of variable x_i.
- $F : X \rightarrow A$ is a function that assigns variables to agents. Each agent controls (the value of) the variables that are assigned to it.
- $C = \{C_1, \ldots, C_m\}$ is now a set of *MABs*. In more detail, each constraint function C_j models a MAB, such that each joint assignment $X_j \in D_{X_j}$ of the agents connected to that MAB becomes an arm of that bandit. Hence, each value $C_j(X_j)$, obtained by choosing $X_j \in D_{X_j}$, is assumed to be drawn from an *unknown*, but *fixed*, distribution, with (unknown) expected value $\mu(X_j)$, and bounded support $[u_{\min}, u_{\max}]$.

Thus, in a MAB-DCOP, as is the case for a general PO-DCOP, the sets A, X, D and F maintain the same semantics as for traditional DCOPs. The constraint functions C_j are then defined as MABs which means that there is no *a priori* knowledge about the constraint functions that govern the agents' interactions, and these interactions are subject to stochasticity. This is a common situation in multiple real world problems. For instance, as we discussed at the beginning of this chapter, in the domain of disaster response, multiple scenarios can be identified in which the agents need to learn the outcome of their decisions (one is presented in Chapter 1).

At the beginning of this chapter, we described a MAB-DCOP as a sequential problem rather than a canonical one shot optimisation problem. Hence, in a MAB-DCOP the agents' goal is to choose an optimal joint assignment $X^*(t) = \langle x_1^*(t), \ldots, x_n^*(t) \rangle$ at each time step t, such that the expected cumulative utility over a finite time horizon T is maximised:

$$\begin{aligned} [X^*(1), \ldots, X^*(T)] &= \underset{X(1),\ldots,X(T)}{\arg\max}\ \mathbb{E}\left[\sum_{t=1}^{T}\sum_{j=1}^{m} C_j\left(X_j(t)\right)\right] \\ &= \underset{X(1),\ldots,X(T)}{\arg\max} \sum_{t=1}^{T}\sum_{j=1}^{m} \mu\left(X_j(t)\right) \end{aligned} \tag{7.1}$$

where $X_j(t)$ is the chosen assignment to X_j at time t.

It should be noted, that if the constraint functions were known and if the level of uncertainty was zero, then a MAB-DCOP would regress to a canonical DCOP. More formally, we can define the following property:

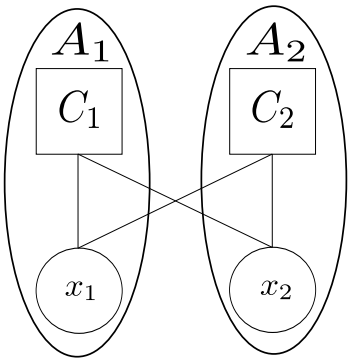


FIGURE 7.1: A factor graph encoding the MAB-DCOP presented in Example 7.1.

Property. A DCOP is a MAB-DCOP in which the outcome of each of the constraint functions in C is known with precision.

To illustrate MAB-DCOPs, consider the following example:

Example 7.1. *Consider a MAB-DCOP with* $A = \{A_1, A_2\}$, $X = \{x_1, x_2\}$, $D = \{\{0,1\},\{0,1\}\}$, *and* $C = \{C_1(x_1,x_2), C_2(x_1,x_2)\}$, *at time* $t = 10$. *Figure 7.1 illustrates the factor graph associated with it. The constraints are defined as follows:*

x_1	x_2	$C_1(x_1,x_2)$	$C_2(x_1,x_2)$	$C(x_1,x_2)$
0	0	$(3.0,1)$	$(2.5,1)$	$(5.2,2)$
0	1	$(5.0,3)$	$(3.2,3)$	$(8.2,7)$
1	0	$(2.0,2)$	$(1.7,1)$	$(3.7,3)$
1	1	$(1.1,2)$	$(4.1,2)$	$(5.2,4)$

Constraint functions are represented as tables, each cell of which has a tuple $(\hat{\mu}(X_j,t), n(X_j,t))$, *where* $\hat{\mu}(X_j,t)$ *is the sample mean of the constraint received for the assignment* C_j *at* t, *and* $n(X_j,t)$ *is the number of times that assignment has been sampled. Given the current sample means, assignment* $(x_1 = 0, x_2 = 1)$ *seems to be optimal. However,* $(x_1 = 0, x_2 = 0)$ *could be the real optimal, since* $C_1(0,1)$ *and* $C_2(0,1)$ *have only been sampled once.*

As this example suggests, to solve MAB-DCOPs, agents need to coordinate over the assignments to variables X at each time step, in order to trade off exploration (reducing uncertainty about the expected utility of each joint assignment) and exploitation (using the joint assignment that is believed to maximise reward).

Similar to standard MABs (see Chapter 2), we can define the regret of a coordination algorithm in a MAB-DCOP as a measure of the performance of a particular algorithm. In more detail, let $P = X^P(1), X^P(2), \ldots$ denote a coordination algorithm that chooses

the joint assignment $X^P(t) = \langle x_1^P(t), \ldots, x_n^P(t) \rangle$ at time t. The regret $R_P(T)$ for T time steps can be formalised as:

$$R_P(T) = T \cdot \max_X \sum_{j=1}^{m} \mu(X_j) - \sum_{t=1}^{T} \sum_{j=1}^{m} \mu\big(X_j^P(t)\big) \tag{7.2}$$

Hence, since we are interested in minimising such regret (i.e. thus generalising the procedure used for canonical MABs), we follow the same procedure used in Chapters 5 and 6 and use the PO-GDL to instantiate new algorithms to solve MAB-DCOPs. To achieve this, as discussed in Chapter 2, a key idea is to use the UCB. For this reason we present in the next section, the way in which we adapt it to a multi-agent setting and incorporate in a commutative semi-ring that can be used to instantiate the PO-GDL.

7.2 An Algorithm for solving MAB-DCOPs based on the PO-GDL

We present in this section a class of algorithms to solve MAB-DCOPs. We name this class of algorithms the MAB-GDL since, as was the case in Chapters 5, and 6, it exploits the PO-GDL framework described in Chapter 4.

In more detail, we present in what follows an optimal algorithm, proceeding over three phases:

Phase 1 - Junction Tree Computation : The *junction tree* (JT) phase produces a junction tree of the factor graph encoding an MAB-DCOP, using the same procedure as the canonical Action GDL algorithm.

Phase 2 - Message Passing : The *message passing* (MP) phase is a re-definition of the canonical GDL message passing phase to the setting in which the agents coordinate to find the non-dominated set of solutions to the cycle-free factor graph computed in the JT phase.

Phase 3 - Value Propagation : The *value-propagation* (VP) phase allows the agents to select a consistent variable assignment, considering that now multiple non-commensurable alternatives exist.

For the sake of solving MAB-DCOPs appropriately, we need an algorithm capable of recovering optimal solutions. Thus, we focus here on instantiating an optimal algorithm which extends the Action GDL algorithm for solving MAB-DCOPs (we obtain the same algorithm as in Chapter 6). For the sake of our exposition, we name this algorithm

HEIST[1]. Nonetheless, by exploiting the PO-GDL framework this algorithm maintains the same key property as the ones described in Chapters 5 and 6. That is to say, depending on the algorithm used on the first phase, different algorithms can be obtained.

In what follows, we describe each of these phases in detail, and then we analyse some of the algorithm's key properties.

7.2.1 Algorithm Description

Our algorithm proceeds in three phases. In what follows, we discuss each of these phases in detail.

7.2.1.1 The Junction Tree Phase

As described in the previous chapters, this phase is used to transform any factor graph encoding a MAB-DCOP into an acyclic factor graph. Various techniques can be used, most of which have been discussed in Chapters 2 and 5.

In this chapter, however, in contrast to the previous chapters, we focus on studying the overall class of algorithms, and do not consider a specific algorithm for phase 1. Nonetheless, to keep our exposition clear, we will assume in the remainder of this chapter that a junction tree algorithm has been used for phase 1 of the PO-GDL algorithms (see Chapters 2, 4 and 6). The following example then shows how the MAB-DCOP described in Example 7.1 is transformed into a junction tree.

Example 7.2. *The MAB-DCOP in Example 7.1 describes a setting with two variables x_1 and x_2, and two constraints C_1 and C_2. This can be encoded in the factor graph depicted in Figure 7.2.*

Now, to compute a junction tree, variables x_1 and x_2 are merged into a new variable x_p, the new RA-DCOP is defined as follows:

$x_p = (x_1, x_2)$	$C_1(x_p)$	$C_2(x_p)$	$C(x_p)$
$a = (0, 0)$	$(3.0, 1)$	$(2.5, 1)$	$(5.5, 2)$
$b = (0, 1)$	$(5.0, 3)$	$(3.2, 3)$	$(8.2, 7)$
$c = (1, 0)$	$(2.0, 2)$	$(1.7, 1)$	$(3.7, 3)$
$d = (1, 1)$	$(1.1, 2)$	$(4.1, 2)$	$(5.2, 4)$

Constraint functions are represented as tables, each cell of which has a tuple $(\hat{\mu}(X_j, t), n(X_j, t))$, where $\hat{\mu}(X_j, t)$ is the sample mean of the constraint received for the assignment C_j at t,

[1]The name comes from the fact that the algorithm coordinates a group of bandits.

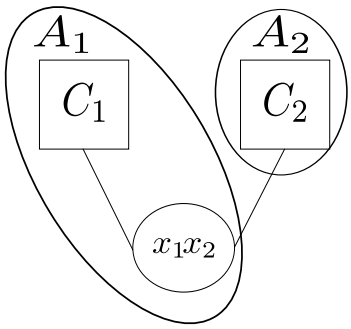


FIGURE 7.2: The junction tree from the factor graph in Figure 7.1

and $n(X_j, t)$ is the number of times that assignment has been sampled. Given the current sample means, assignment $x_p = b = (0, 1)$ seems to be optimal. However, $x_p = a = (0, 0)$ could be the real optimal, since $C_1(a)$ and $C_2(a)$ have only been sampled twice.

7.2.1.2 The Message Passing Phase

In this section, we present our message passing phase for solving MAB-DCOPs with asymptotically optimal regret. Using this algorithm, agents coordinate at each time step to identify the best joint assignment to variables X, i.e. the arms that should be pulled on the local MABs (functions C_j) to minimise regret over time horizon T.

In more detail, at each time step t, the algorithm uses a PO-GDL message passing phase to find the joint assignment $X^*(t)$ that maximises the upper confidence bound (UCB) (see Chapter 2) on the received utility.

The formulation of this UCB is different from the UCB given in Equation 2.5—in fact, it is a generalisation—since the objective in a MAB-DCOP is to maximise the sum of the rewards of *multiple* local MABs, instead of a single MAB:

$$X^*(t) = \arg\max_X \left[\sum_{j=1}^{m} \hat{\mu}(X_j, t) + \sqrt{2 \ln t \sum_{j=1}^{m} \frac{(u_{\text{range}})^2}{n(X_j, t)}} \right] \tag{7.3}$$

Here, $\hat{\mu}(X_j, t)$ is the sample mean at time t of the utility obtained by assignment X_j from MAB C_j, $n(X_j, t)$ is the number of times a specific assignment to X_j was made, and $u_{\text{range}} = u_{\max} - u_{\min}$.

Example 7.3. *In Example 7.1, $X^*(10) = (x_1 = 0, x_2 = 1)$, with a UCB equal to $8.2 + \sqrt{2 \ln(10) \left(\frac{1}{3} + \frac{1}{3}\right)}$, assuming $u_{\text{range}} = 1$.*

Algorithm 5 The HEIST message passing algorithm for variable x_i

1: $t_{\min} := \max_{k \in [1,m]} |D_{X_k}|$ ▷ Wait for initial samples (line 7)
2: **for** $t = t_{\min} + 1$ to T **do** ▷ For each joint pull
3: $Z_i(x_i) = \text{GDL\_VARIABLE}(i)$ ▷ See Algorithm 3
4: Set $x_i^*(t) := \arg\max_{x_i} \left[\max_{(\hat{\mu}, b^2) \in Z_i(x_i)} (\hat{\mu} + b) \right]$
5: Send message $\mathsf{sample}(x_i^*(t))$ to all $C_j : j \in \mathcal{M}(i)$
6: **end for**

Calculating joint action $X^*(t)$ in Equation 7.3 is not trivial. For instance, this problem cannot be solved using a DCOP algorithm, since the objective function is not decomposable into a sum of factors (i.e. it is non-linear). As a result, the application of a canonical DCOP algorithm to this problem can lead to sub-optimality (Léauté and Faltings, 2011) (which we will show in the empirical evaluation).

In contrast, by applying the methodology described in Chapter 4, we can instantiate the PO-GDL to derive a class of algorithms with a number of properties, such as the optimality over acyclic factor graphs (which we will shortly see) which is guaranteed to preserve the joint assignment with the optimal UCB. To achieve this, we describe the commutative semi-ring characterising the objective function of a MAB-DCOP to minimise regret (Equation 7.3). Note that in contrast to the semi-rings defined in Chapters 5 and 6, the semi-ring defined here is not a property of the objective function to optimise. Rather, we use it here as an algebraic solution to define algorithms that can solve MAB-DCOPs by providing optimal regret. This semi-ring, which we define as the UCB-GDL semi-ring, is defined as follows:

Definition 7.2. The UCB-GDL semi-ring is a semi-ring $\langle R, \max_{\succ}, + \rangle$ such that:

- $R = \mathcal{P}(\mathbb{R} \times \mathbb{R})$ is a *set of sets* of tuples as described in Definition 4.2, which have the same signature as the tuples output by functions C_j—the first element of each tuple is a sample mean $\hat{\mu}$, and the second is a squared bound b^2. The identity elements are $\mathbf{0} = \{(-\infty, -\infty)\}$ and $\mathbf{1} = \{(0, 0)\}$.
- $\max_{\succ}$ is an operator that takes multiple sets $S_1, \ldots, S_k \subseteq R$ as input and outputs a set S' such that:

$$S' = \max_{\succ}(S) = \left\{ s \in \bigcup_{i=1}^{k} S_i \mid \forall s' \in \bigcup_{i=1}^{k} S_i : s' \not\succ s \right\}$$

  Operator $\max_{\succ}$ filters out so-called *dominated* tuples from sets $S_1, \ldots, S_k$—those that cannot maximise the global UCB. Domination is formally defined by binary

Algorithm 6 The HEIST message passing algorithm for function C_j. Line numbering continued from Algorithm 5.

```
7: for each X'_j ∈ D_{X_j}, sample C_j(X'_j) once
8: t_min := max_{k∈[1,m]} |D_{X_k}|
9: for t = t_min + 1 to T do                          ▷ For each joint pull
10:    execute GDL_FUNCTION(j)                        ▷ See Algorithm 4
11:    Wait for sample(x*_i(t)) messages from all x_i : i ∈ N(j)
12:    Pull arm X*_j(t) = {x*_i(t) | i ∈ N(j)}
13:    Update sample mean μ̂(X*_j, t) and sample count n(X*_j, t)
14: end for
```

operator $\succ$ such that $(\hat{\mu}_1, b_1^2) \succ (\hat{\mu}_2, b_2^2)$ iff:

$$
\begin{aligned}
&(\hat{\mu}_1 - \hat{\mu}_2 \geq b_2 - b_1) \wedge \\
&\left(\hat{\mu}_1 - \hat{\mu}_2 \geq \sqrt{b_2^2 + u_{\text{range}}^2 2 \ln t} - \sqrt{b_1^2 + (u_{\text{range}})^2 2 \ln t}\right) \wedge \\
&\left(\hat{\mu}_1 - \hat{\mu}_2 \geq \sqrt{b_2^2 + u_{\text{range}}^2 \frac{2 \ln t}{t}} - \sqrt{b_1^2 + (u_{\text{range}})^2 \frac{2 \ln t}{t}}\right)
\end{aligned}
$$

- $+$ is a binary operator such that if $S_1, S_2 \in R$, then $S_1 + S_2 = \{(\hat{\mu}_1 + \hat{\mu}_2, b_1^2 + b_2^2) \mid \forall(\hat{\mu}_1, b_1^2) \in S_1, \forall(\hat{\mu}_2, b_2^2) \in S_2\}$. The key idea here is to define an operator that aggregates the parameters necessary to update the global UCB (see Equation 7.3). More specifically, the estimates $\hat{\mu}$ of the functions in C and the bounds b_j on the uncertainty over these estimates are added together in a way that is consistent with the global UCB.

The $\succ$ operator presented in the definition above, is derived by algebraic manipulation. In Section 7.2.2, we will present the way the operator is derived (see the proof of Theorem 7.3).

Using this semi-ring, we can instantiate our algorithm, which proceeds as any canonical GDL algorithm. It is split up into two algorithms, one for variables (Algorithm 2.5) which extends Algorithm 2.1, and one for functions (Algorithm 2.6) which extends Algorithm 2.2. The main difference is that the propagated messages here, contain information about the samples $\hat{\mu}$ and the bounds b_j. The reason for propagating bounds is that they are used to calculate the number of pulls of multiple arms when calculating function to variable messages $R_{j \to i}(x_i)(x_i)$ (using $\max_{\succ}$ from Definition 7.2). In addition, similarly to the UCB algorithm presented in Chapter 2, which pulls each arm once at the start, before communication commences, functions first sample the utility for their local domains (line 7) which takes $\max_{k \in [1,m]} |D_{X_k}|$ time steps.

Note, that constraint functions are now vector functions (as per definition of a PO-DCOP) defined over tuples of the form: $C_j(X_j, t) = (\hat{\mu}(X_j, t), b(X_j, t)^2)$. Here, $\hat{\mu}(X_j, t)$

is the sample mean of assignment $X_j \in D_{X_j}$ at time t and $b(X_j, t) = u_{\text{range}}\sqrt{2 \ln t / n(X_j, t)}$ is its (local) upper confidence bound t (cf. the two terms in Equation 2.5).

As a consequence of the non-linearity of the objective function in Equation 7.3, choosing the assignment that maximises the sum of UCBs on local utility, does not (necessarily) produce optimality of the UCB on the sum of local utilities. Thus, we cannot simply discard one tuple $(\hat{\mu}_1, b_1^2)$ in favour of tuple $(\hat{\mu}_2, b_2^2)$ if $\hat{\mu}_1 + b_1 \geq \hat{\mu}_2 + b_2$. This problem is addressed by the definition of the $\max_{\succ}$ operator, which is designed to discard only those tuples that are *guaranteed* to lead to global sub-optimality. As a result, it imposes a partial order over tuples to preserve the tuple that produces global optimality.

Executing the GDL algorithm on the UCB-GDL semi-ring yields the marginal function $Z_i(x_i)$ for each variable x_i (line 3). It can be proved that for each assignment $x_i \in D_i$, the set $Z_i(x_i)$ contains a tuple $(\hat{\mu}, b^2)$, such that $\hat{\mu} + b$ is the maximum achievable global UCB given that assignment (Theorem 7.3). Using this marginal function $Z_i(x_i)$, the second phase of our algorithm first computes the maximum UCB for each assignment x_i (line 4, expression between brackets) and then selects the assignment with the maximum UCB (remainder of line 4). Then, in line 5, each variable informs adjacent functions of its assignment, after which all functions sample assignments (line 12).

The following example illustrates the behaviour of our algorithm:

Example 7.4. *The following example demonstrates the operation of the message passing phase on a single time step ($t = 10$) of the MAB-DCOP from Example 7.1. Let $c = 2ln(10)$, and $u_{\text{range}} = 1$. In order to keep the exposition as clear as possible, we consider a simple message passing schedule, where the computation starts at the leaf nodes and goes up through the different nodes of the graph. The computation begins at time step $t = 0$, nodes C_1 and C_2 both have a single edge (i.e. they are leaves of the cycle free factor graph) and therefore send a message first:*

$R_{1\to p}(x_p) =$

x_p	C
$a = (0,0)$	$[3.0, c]$
$b = (0,1)$	$[5.0, \frac{c}{3}]$
$c = (1,0)$	$[2.0, \frac{c}{2}]$
$d = (1,1)$	$[1.1, \frac{c}{2}]$

$R_{2\to p}(x_p) =$

x_p	C
$a = (0,0)$	$[2.5, c]$
$b = (0,1)$	$[3.2, \frac{c}{3}]$
$c = (1,0)$	$[1.7, c]$
$d = (1,1)$	$[4.1, \frac{c}{2}]$

Both C_1 and C_2 computes the multi-objective marginal function of variable x_p, as detailed above. At time step $t = 1$, x_p receive the messages and compute the next two messages:

$$Q_{p\to 1}(x_p) = \begin{array}{c|c} x_p & C \\ \hline a=(0,0) & [2.5, c] \\ b=(0,1) & [3.2, \frac{c}{3}] \\ c=(1,0) & [1.7, c] \\ d=(1,1) & [4.1, \frac{c}{2}] \end{array} \quad Q_{p\to 2}(x_p) = \begin{array}{c|c} x_p & C \\ \hline a=(0,0) & [3.0, c] \\ b=(0,1) & [5.0, \frac{c}{3}] \\ c=(1,0) & [2.0, \frac{c}{2}] \\ d=(1,1) & [1.1, \frac{c}{2}] \end{array}$$

After these messages have been sent, the MS phase terminates. Finally, variable x_p can compute its non-dominated values:

$$ND_{X(10)} = \max_{\succ} \left\{ \begin{array}{c|c} x_p & C \\ \hline a=(0,0) & [5.5, \dot{c(1+1)}] \\ b=(0,1) & [8.2, \dot{c(\frac{1}{3}+\frac{1}{3})}] \\ c=(1,0) & [3.7, \dot{c(1+\frac{1}{2})}] \\ d=(1,1) & [5.2, \dot{c(\frac{1}{2}+\frac{1}{2})}] \end{array} \right\}$$
$$= \{[8.2, \dot{c(\frac{1}{3}+\frac{1}{3})}]\}$$

Such values correspond to the non dominated solutions of the problem $O_{X(10)} = \{x_p = (0,1) = b\}$.

7.2.1.3 The Value-Propagation Phase

The value-propagation phase operates on the cycle-free factor graph computed in the first phase. Variables and function nodes in the factor graph coordinate to select a consistent variable assignment. Since, MAB-DCOPs are PO-DCOPs, we can use the value-propagation described in Chapter 4, by instantiating the $\succ$ relation using the dominance condition defined in the previous section.

However, in contrast to the example presented in Chapter 5 and similarly to the one presented in Chapter 6, since the solution of Example 7.1 is unique, the value propagation phase is not necessary in this case.

7.2.2 Theoretical Analysis

The key focus of our theoretical analysis is demonstrating that when solving MAB-DCOPs using a GDL-based algorithm we can achieve optimal regret. To demonstrate this, we first state the following result, similar to those presented in Chapters 5 and 6:

Theorem 7.3. *If the factor graph is acyclic and the stopping criterion is chosen such that the GDL message passing phase is run for a number of iterations that is equal to the*

diameter of the factor graph, the following equation holds for each $t > \max_{k\in[1,m]} |D_{X_k}|$*:*

$$\max_{(\hat{\mu},b^2)\in Z_i(x_i)} (\hat{\mu}+b) = \max_{X\setminus x_i} \left[\sum_{j=1}^{m} \hat{\mu}(X_j,t) + \sqrt{2\ln t \sum_{j=1}^{m} \frac{(u_{\text{range}})^2}{n(X_j,t)}} \right]$$

Proof. We first show that operator $\max_{\succ}$ filters out tuples that cannot maximise the global UCB. In particular, if there is at least one incoming message, then $(\hat{\mu}_2, b_2^2)$ is dominated by $(\hat{\mu}_1, b_1^2)$ if and only if for any incoming tuple $(\hat{\mu}_3, b_3^2)$, $\hat{\mu}_1+\hat{\mu}_3+\sqrt{b_1+b_3} > \hat{\mu}_2+\hat{\mu}_3+\sqrt{b_2+b_3}$ holds. This implies that $\hat{\mu}_1-\hat{\mu}_2 > \sqrt{b_2+b_3}-\sqrt{b_1+b_3}$. By definition, $b_3 = u_{\text{range}}\sqrt{\frac{2\ln t}{n_3(t)}}$ at time t, where $n_3(t)$ is the number of samples included in $\hat{\mu}_3$. Note that $1 \leq n_3(t) \leq t$, and can take any arbitrary value within this interval. That is, $u_{\text{range}}^2 \frac{2\ln t}{t} \leq b_3^2 \leq u_{\text{range}}^2 2\ln t$. This implies that if at least one among the second or the third clause in the definition does not hold, then tuple $(\hat{\mu}_2, b_2^2)$ cannot be discarded (i.e. it is not dominated by tuple $(\hat{\mu}_1, b_1^2)$. In addition, if there are no incoming messages, then breaking the first clause indicates that $(\hat{\mu}_2, b_2^2)$ cannot be discarded either. That is, $(\hat{\mu}_2, b_2^2)$ is dominated by $(\hat{\mu}_1, b_1^2)$ iff all the clauses hold.

The proof of the claim that the MAB-GDL message passing phase yields $Z_i(x_i)$, the maximum marginal UCB for each assignment, follows a similar argument from that of Theorem 4.3. □

Put differently, Theorem 7.3 states that, after the initial pulls, set $Z_i(x_i)$ contains the tuple that yields the *marginal maximum UCB* that can be obtained for each assignment to x_i, $i \in [1,n]$. As a direct consequence, line 12 pulls the arm on each function with the highest overall UCB (Equation 7.3). This observation leads to the following theorem:

Theorem 7.4. *Suppose the factor graph is acyclic and the stopping criterion is chosen such that MAB-GDL message passing phase is run for a number of iterations that is equal to the diameter of the factor graph at every time* t*. Let* $X^{\mathbb{E}} = \arg\max_X \sum_{j=1}^{m} \mu(X_j)$ *be the joint assignment that maximises the (unknown) expected utility. Let* $d(X) = \mu\left(X^{\mathbb{E}}\right) - \mu(X)$ *denote the difference between the expected utility of the optimal action* $X^{\mathbb{E}}$ *and that of a particular joint action* X*. Given this, the cumulative regret* $R_{\text{HEIST}}(T)$ *of algorithm after* T *time steps is at most:*

$$\sum_{X\neq X^{\mathbb{E}}} \frac{u_{\text{range}} 8\ln T}{d(X)} + \left(1+\frac{\pi^2}{3}\right) \sum_{X\neq X^{\mathbb{E}}} d(X) \tag{7.4}$$

Proof. The demonstration proceeds by calculating an upper bound on the regret (see Equation 7.2) of HEIST. We know, by theorem 7.3, that after the MAB-GDL message

passing phase has converged, the algorithm chooses the joint assignment with the maximum UCB at each time $t \leq T$. More formally, suppose that at each time t, HEIST chooses joint assignment $X^*(t) = \langle x_1^*(t), \ldots, x_n^*(t) \rangle$.

To determine the bound, the idea is to estimate the expected number of times $X^*(t) \neq X^{\mathbb{E}}$ is chosen. Formally, let $N_T(X)$ denote the number of times the algorithm chooses a suboptimal joint assignment $X \neq X^{\mathbb{E}}$ before the last time step T. By estimating $N_T(X)$, we can estimate the number of times HEIST chooses a suboptimal joint assignment, and thus, derive a bound on its regret. Such bound is defined as follows:

$$R_{\text{HEIST}}(T) \leq \sum_{X \in \prod D_i} N_T(X) d(X) \tag{7.5}$$

Now, we denote the expected number of times $X^*(t) \neq X^{\mathbb{E}}$ is chosen as $\mathbb{E}[N_T(X)]$. Intuitively, we can bound $\mathbb{E}[N_T(X)]$ by calculating the probability that the algorithm always chooses a sub-optimal assignment for T time steps (except the first k steps, where the algorithm pulls each of the k arms as shown in Algorithm 6). Formally, this bound is defined as follows:

$$\mathbb{E}[N_T(X)] \leq k + \sum_{t=1}^{T} P\left(X^*(t) = X, X \neq X^{\mathbb{E}}, N_{t-1}(X) \geq k\right) \tag{7.6}$$

Now, the latter term can be further upper bounded by noticing that the probability that HEIST chooses $X^*(t) \neq X^{\mathbb{E}}$ can be bounded by the probability that $X^*(t)$ has a higher UCB than $X^{\mathbb{E}}$ for each time step t. In other words, the second term of Equation 7.6 is further bounded by the probability that $\hat{\mu}(X(t), t) + b(X(t), t) \geq \hat{\mu}(X^{\mathbb{E}}, t) + b(X^{\mathbb{E}}, t)$. More formally, this probability is defined as follows:

$$\sum_{t=1}^{T} P\left(\hat{\mu}(X(t), t) + b(X(t), t) \geq \hat{\mu}(X^{\mathbb{E}}, t) + b(X^{\mathbb{E}}, t), N_{t-1}(X) \geq k\right) \tag{7.7}$$

where $b(X, t) = \sqrt{2 \ln(t) \sum_{j=1}^{m} \frac{(u_{\text{range}})^2}{n(X_j, t)}}$. This can be further bounded by the probability that the UCB of each possible assignment $X^*(t)$ is higher than the UCB of each possible $X^{\mathbb{E}}(t')$ with $t \neq t'$. This probability is defined as follows:

$$\sum_{t=1}^{T} \sum_{s_j=k}^{t} \sum_{s=k}^{t} P\left(\hat{\mu}(X(s_j), s_j) + b(X(s_j), s_j) \geq \hat{\mu}(X^{\mathbb{E}}, s) + b(X^{\mathbb{E}}, s)\right) \tag{7.8}$$

Now, by using similar argument to that of Theorem 1 in (Auer et al., 2002), it is true that if $\hat{\mu}(X(s_j), s_j) + b\left(X\left(s_j\right), s_j\right) \geq \hat{\mu}(X^{\mathbb{E}}, s) + b(X^{\mathbb{E}}, s)$ holds then at least one of the following must hold:

1. $\hat{\mu}\left(X^{\mathbb{E}}, s\right) + b\left(X^{\mathbb{E}}, s\right) \leq \mu\left(X^{\mathbb{E}}, s\right)$.
2. $\mu\left(X(s_j), s_j\right) \leq \hat{\mu}\left(X(s_j), s_j\right) + b\left(X(s_j), s_j\right)$.
3. $\mu\left(X^{\mathbb{E}}, s\right) - \mu\left(X(s_j), s_j\right) \leq 2b(X(s_j), s_j)$.

By using McDiarmid's inequality (McDiarmid, 1989), we can show that both (1) and (2) hold with probability t^{-4}, and if $k \geq \lceil \frac{u_{\text{range}} 8 \ln T}{d(X)} \rceil$, then (3) does not hold. This implies that:

$$\mathbb{E}\left[N_T\left(X\right)\right] \leq k + \sum_{t=1}^{T} \sum_{s=1}^{t} \sum_{s_j=1}^{t} 2t^{-4} \leq k + \frac{\pi^2}{3} \tag{7.9}$$

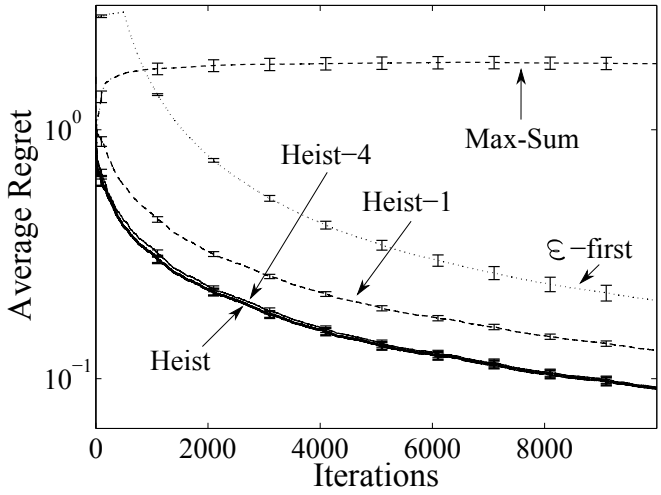
for any $k \geq \lceil \frac{u_{\text{range}} 8 \ln T}{d_X} \rceil$. The last inequality is obtained from the Riemann Zeta Function (Edwards, 2001) for value of 2. Finally, substituting this into Equation 7.5 concludes the proof. □

Based on this result, we can show that HEIST provides asymptotically optimal regret bounds, by comparing against best achievable regret:

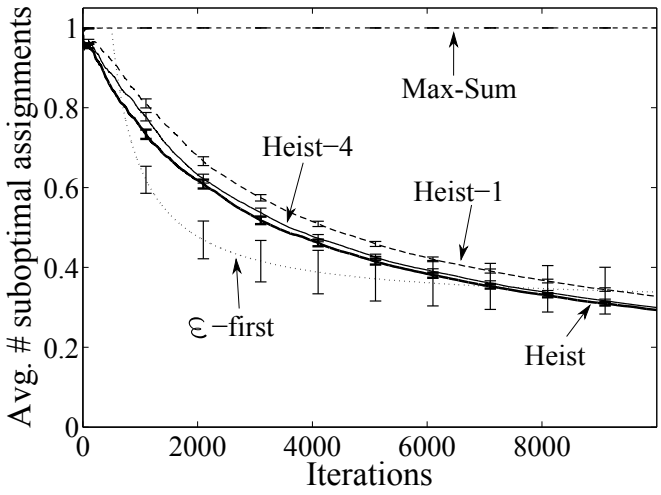
Theorem 7.5. *For any algorithm, there exists a constant $C \geq 0$, and a particular instance of the MAB-DCOP problem, such that the regret of that algorithm within that particular problem is at least $C \ln T$.*

Proof. We can reduce all standard MAB problems to a MAB-DCOP with $m = 1$. According to Lai and Robbins (1985), the best possible regret that an algorithm can achieve on standard MABs is $C \ln T$. Therefore, if there is an algorithm for MAB-DCOPs that provides better regret than $C \ln T$, then it also provides better regret bounds for standard MABs. □

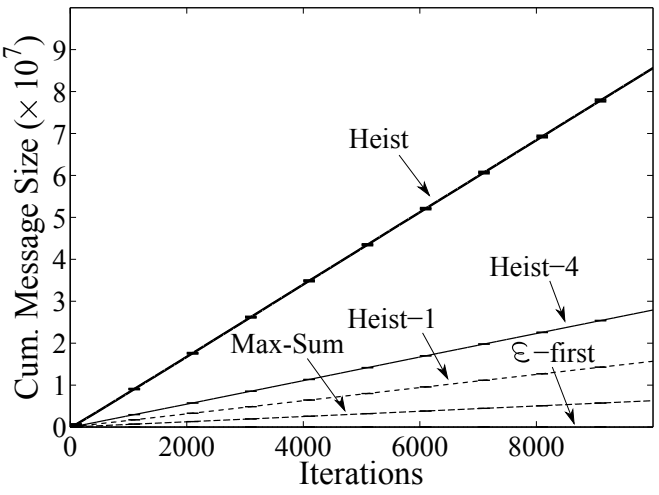
Thus, the regret bound of HEIST (Equation 7.4) only differs from the best possible with a constant factor.



(a) Regret

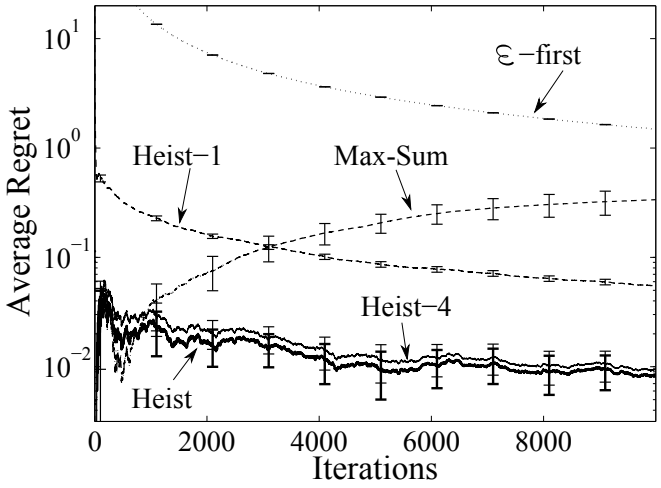


(b) Suboptimal joint assignments

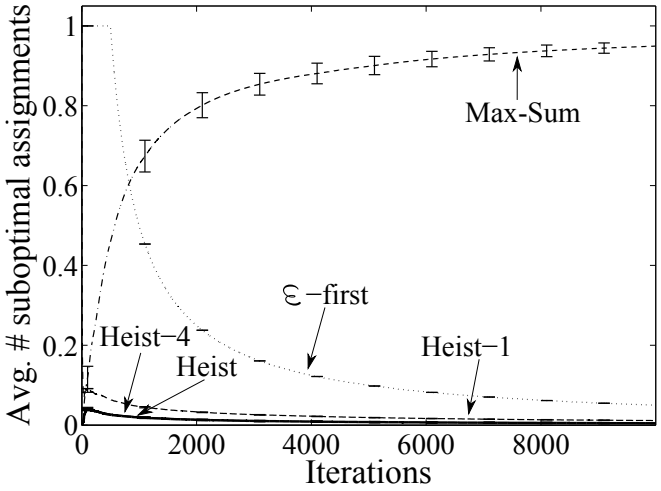


(c) Message size

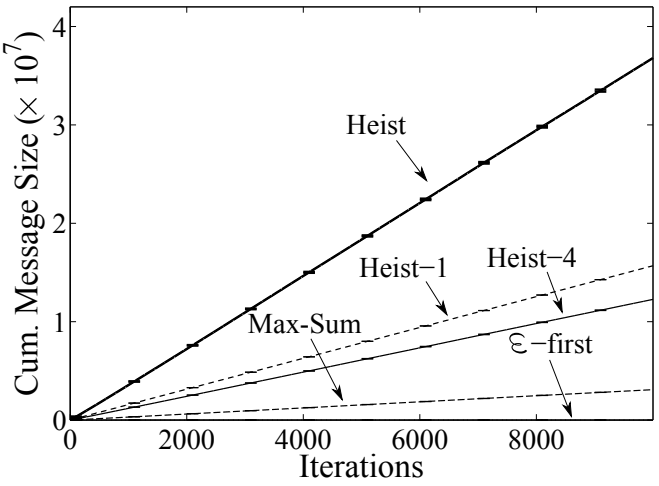
FIGURE 7.3: Empirical results for $\mu_{\max} = 1$



(a) Regret



(b) Suboptimal joint assignments



(c) Message size

FIGURE 7.4: Empirical results for $\mu_{\max} = 10$

7.3 Empirical Evaluation

In the previous section, we proved that the regret achieved by HEIST is guaranteed to be a constant factor away from the optimal. However, further empirical analysis is needed to gauge HEIST's practical performance, in terms of solution quality as well as communication and computation overhead. Moreover, such analysis can focus on the algorithm's performance when the MAB-GDL phase is not run until convergence, one of the conditions for optimality in Theorem 7.4. Instead, the number of iterations c of the GDL phase can be a parameter for tuning the trade off between solution quality and overhead (i.e. computation and communication).

Therefore, in this section, we benchmark several versions of HEIST against existing approaches, taken from the state of the art in the MAB and DCOP literature. Specifically, we compare the algorithm against the following approaches:

HEIST-c a version of HEIST where the GDL message passing phase is run for c iterations. When taking joint actions is cheap compared to communication, or when action is required before the GDL message passing phase is able to converge, c can be set to a value smaller than the diameter of the factor graph. In this case, optimality is no longer guaranteed, but it can lead to a good trade off between communication and solution quality.

ε-first an algorithm that samples from the utility functions (exploration) for the first εT time steps, and picks the one that is believed to be optimal (exploitation) for the remaining $(1-\varepsilon)T$ time steps. At the start of the exploitation phase, this algorithm runs a standard DCOP algorithm (max-sum) once to find the joint assignment that maximises the sum of the sample means of the local utilities. Using a DCOP algorithm in this way is equivalent to using E-DPOP (Léauté and Faltings, 2009) on a problem where each local assignment X_j for $j \in [1, m]$ is modelled by a single random variable. Thus, ε-first can be considered as E-DPOP applied to a MAB-DCOP. To perform well, ε needs to be tuned for each problem instance (Auer et al., 2002; Sutton and Barto, 1998). After initial tests, we found that $\varepsilon = 0.02$ leads to good performance for the problems described below.

Monolithic UCB the (centralised) UCB algorithm (see Section 2.4.3) that considers a MAB-DCOP as a single "monolithic" MAB with $\prod_{i=1}^{n} |D_i|$ arms.

Max-Sum a standard DCOP algorithm, applied to a DCOP in which the objective is to compute $X^+(t)$ at every $t \in [1, T]$, such that:

$$X^+(t) = \arg\max_{X} \sum_{j=1}^{m} \left[\hat{\mu}(X_j, t) + \sqrt{\frac{(u_{\text{range}})^2 2 \ln t}{n(X_j, t)}} \right] \tag{7.10}$$

By solving this DCOP, the sum of UCBs on local utilities is maximised, instead of the UCB on the sum of utilities (Equation 7.3). Since the latter is clearly not linear, Equations 7.3 and 7.10 are not equivalent (except for $m = 1$). As a result, this decomposition leads to loss of optimality in the sense of Theorem 7.5. Léauté and Faltings (2011) observed a similar result for their (incomplete) algorithm when using a linear decomposition to maximise non-linear objective functions.

The ε-first and max-sum algorithms are included to demonstrate that standard DCOP algorithms are unsuitable for solving MAB-DCOPs, while the Monolithic UCB algorithm is included to demonstrate the need to exploit the factorisability of a MAB-DCOP.

We randomly generated MAB-DCOP instances that can be encoded as acyclic factor graphs, with $n = 15$ and $m = 14$, and $|D_i| = 3$. Each constraint function $C_j(X_j)$ is governed by a set of normal distributions, one for each assignment to X_j. In more detail, $C_j(X_j) \sim \mathcal{N}(\mu(X_j), \sigma^2(X_j))$, where $\mu(X_j)$ and $\sigma^2(X_j)$ are uniformly drawn from intervals $[0, \mu_{\max}]$ and $[0, 1]$ respectively. The idea here follows from Chapter 6 and consists of creating a controllable environment in which we can reproduce several level of uncertainty to cover a vast majority of the situations that could arise in the real world. Parameter $\mu_{\max}$ is used to control the relative importance that needs to be given to exploration and exploitation. When $\mu_{\max}$ is decreased, the received utilities become more noisy and the balance should be shifted towards exploration. Conversely, when $\mu_{\max}$ is increased, the optimal joint action becomes more easily identifiable, and agents start exploitation quite early on. For our experiments, we chose $\mu_{\max} = 1$ and $\mu_{\max} = 10$. These values were chosen during initial calibration to yield two sets of difficult problems, which simultaneously demonstrate the difference in required emphasis between exploration and exploitation. The idea of controlling the balance between exploration and exploitation allows us, again, to reproduce a variety of outcomes of agents' decisions that might arise in the real world. For instance, while mapping the scene of a disaster searching for casualties, UAVs might not know with precision the number of casualties within each area. More specifically, each area might have a different estimate of such number, with a different level of uncertainty. Hence, by designing tests with a varying level of uncertainty, we are able to design problems capable of encompassing all such settings.

The results are shown in Figures 7.3 and 7.4 for $\mu_{\max} = 1$ and $\mu_{\max} = 10$ respectively. Each algorithm was run 64 times on both problem classes to obtain statistically significant results. Error bars indicate the standard error of the mean. Monolithic UCB is omitted from all figures, because its regret converged approximately a factor of 3^{15} slower than Heist. This was expected, as it regards the problem as a MAB with 3^{15} arms, instead of 14 MABs with 9 arms each.

Now, Figures 7.3(a) and 7.4(a) show the average regret for the remaining algorithms, while Figures 7.3(b) and 7.4(b) show the number of average suboptimal assignments. As can be observed from these figures, HEIST and HEIST-4 outperform all others in terms of regret (up to 1.5 orders of magnitude for $\mu_{\max} = 10$). We found that for $c \geq 8$, the performance of HEIST (which was run for 30 iterations to ensure convergence) and HEIST-c coincide, indicating that the algorithm performs well even when conditions of Theorem 7.3 are not met. Max-sum—and indeed any algorithm that solves Equation 7.10—is clearly suboptimal, as its regret does not converge to zero, and it consistently produces suboptimal assignments. The fact the regret of max-sum *increases* is counter intuitive. However, additional experimentation showed that for smaller problem instances, the difference between HEIST and max-sum is much smaller, and their performance often coincides for problems with $m < 5$, while for $m = 50$, the difference in regret at $T = 10000$ was found to be more than 3 orders of magnitude. This leads us to conclude that for larger problems, the non-linearity of Equation 7.3 is more pronounced, increasing the difference between the regret associated with assignments $X^+(t)$ and $X^*(t)$. The regret of the second DCOP based technique, ε-first, does converge to zero, but at a much slower pace than HEIST. Based on these results, we can conclude that both DCOP techniques are unsuitable for solving MAB-DCOPs.

Focusing on communication overhead, Figures 7.3(c) and 7.4(c) show the cumulative message size expressed as the number of floating point values exchanged between the agents. Compared to ε-first (which needs a negligible number of messages to coordinate once) and max-sum (which exchanges scalars, instead of sets of tuples), HEIST requires more communication. However, a good balance can be struck between solution quality and communication by reducing c to 4. Moreover, note that HEIST requires each agent to exchange only 600 values per iteration of the MAB–DCOP, a value that is well within the capabilities of bandwidth constrained embedded agents. Finally, the computation required by HEIST to solve problem instances with $n = 50$ and $T = 20000$ never exceeded 4 hours on a standard desktop PC, which is less than 200ms per agent per iteration. Again, this is well within the reach of embedded agents.

7.4 Summary

This chapter presented a new class of problems and solution techniques to address coordination problems involving uncertainty and lack of prior information on the outcome of the agents' decisions. More specifically, in Section 7.1, we proposed the MAB-DCOP framework, a specialisation of the PO-DCOP framework defined in Chapter 4. The key idea of a MAB-DCOP is that each partially ordered constraint function of a PO-DCOP becomes a MAB. Hence, we propose here a further class of problems that meet the complex interaction requirement defined in Chapter 1, in addition to those presented

in Chapters 5 and 6. Furthermore, the MAB-DCOP framework satisfies the generality requirement, since it can represent any type of problem where the agents need to coordinate while learning online the outcome of their decisions.

We then presented, in Section 7.2, a new set of solution techniques for solving MAB-DCOPs. To derive these techniques, we specialised the PO-GDL and derived a new commutative semi-ring based on a generalisation of the UCB (Equation 7.3) in which multiple agents need to coordinate their arm selection (following the MAB terminology defined in Chapter 2). By using this semi-ring, we can then instantiate the message passing algorithms presented in Chapter 2 (Algorithms 2.1 and 2.2) to solve MAB-DCOPs. Hence, as discussed in Chapter 4, we are able to instantiate optimal and bounded approximate decentralised algorithms that meet the requirements of robustness and quality guarantees defined in Chapter 1. In so doing, a key contribution of this chapter was presented in Section 7.2.2 where we demonstrate that using this class of algorithm yields optimal regret (Theorem 7.5).

In Section 7.3, we then presented an empirical analysis to gauge the practical performance of one instantiation of the previous class of algorithms (which we refer to as Heist) in terms of solution quality, as well as communication and computation overhead. Such analysis focused on the algorithm's performance when the MAB-GDL phase is not run until convergence, a situation which is most likely to happen in applied domains such as disaster response. Our results show that the algorithm was capable of achieving a reasonable trade off between the quality of the solutions recovered and the amount of computation and communication necessary to get them. Hence, these results demonstrated that the algorithm positions itself well with respect to the resource awareness requirement defined in Chapter 1. These results were then evaluated on instances of up to 50 agents. Therefore, the algorithm also meets the scalability requirement.

Chapter 8

Conclusions and Future Work

In this chapter, we present an overview of the contributions of this thesis towards the two research objectives presented in Chapter 1. The first objective was to design effective coordination algorithms to enhance the decision making process of a team of agents cooperating to solve a problem. The second objective was to understand the requirements necessary to coordinate these agents in the real world by deploying state-of-the-art coordination algorithms to address real world coordination problems.

In light of this, we discuss, in Section 8.1, the various achievements that we were able to make towards this end. More specifically, we discuss the extent to which each contribution that we were able to make has satisfied the design requirements laid down in Chapter 1. In Section 8.2, we then go on to discuss various potential lines of research that could be pursued as a continuation of this work.

8.1 Summary of Results

A key challenge for modern computer science is the development of technologies that allow interacting computer systems, typically referred as agents, to coordinate their decisions whilst operating in an environment with minimal human intervention. The key idea is that such coordinated behaviour allows each agent to improve its decision making capabilities by taking into account what the remaining agents intend to do.

For this reason, literature that sits at the nexus of artificial intelligence, machine learning and robotics has focused on designing, implementing and deploying novel coordination algorithms capable of improving the performance of such a multi-agent system. Given this background, the work presented in this thesis studied such coordination algorithms and designed new ones to further reduce the gap between the current decision making

capabilities of the agents and those that are actually required to deploy these agents in the real world.

Now, there exist numerous application domains in which adopting coordinated behaviour can improve the agents' overall effectiveness. Examples include coordinating generators in the smart grid (Miller et al., 2012), wireless sensor networks (Rogers et al., 2009) and scheduling multi-processor jobs (Sultanik et al., 2007). Among these domains, this thesis focused on that of disaster response as it involves a range of challenges associated with coordination in complex domains (Chapter 2). In particular, we focused on problems related to the deployment of robotic agents, such as unmanned aerial or ground vehicles (UAVs and UGVs respectively), to achieve situational awareness at the scene of a disaster. Specifically, we described various typologies of unmanned aerial vehicles (UAVs) which constitute the type of robotic agent over which we deployed our coordination algorithms. By analysing these problems and vehicles, we were able to identify the key requirements that need to be satisfied; namely those of *generality*, *scalability*, *robustness*, *resource awareness*, *performance guarantees* and *complex interactions*.

Given these requirements, we moved on to study the degree to which existing algorithms were capable of satisfying them. Initially, we analysed practical approaches (i.e. those that have been deployed in the real world). These are essentially distributed negotiation protocols, in which the agents share their decisions and each agent makes a decision in response to the decisions of the other agents. In so doing, however, the level of performance that these agents can achieve is often low (see Chapter 2 for more details). Hence, the performance guarantee requirement is not met. Moreover, these algorithms often only allow the agents to coordinate over standard decisions defined over one single criteria such as the importance of completing a task or quality of observing a certain area. Thus, the complex interaction requirement is also not met.

In light of this, we moved to study theoretical approaches, where a large body of literature exists, which was focused primarily on deriving new problems capable of modelling sophisticated interactions and on algorithms capable of achieving a high level of performance. We focused our analysis on two key frameworks. First, we studied distributed constraint optimisation problems (DCOPs). We focused on this framework because it has been shown to constitute a reasonable compromise between the way it can encompass a real world problem and the computational complexity that it requires to solve it. In particular, we argued that the DCOP framework positions itself well within the requirements of generality, scalability, resource awareness and robustness. Second, we discussed the generalised distributive law framework (the GDL). We focused on this framework because it is well known for its flexibility. In more detail, the GDL framework can be used to derive optimal (DPOP, Action GDL), bounded approximate (bounded max-sum) and approximate (max-sum) algorithms. These algorithms were shown to fully

satisfy requirements such as scalability and generality and to position themselves well with respect to robustness and resource awareness.

Given these properties, both DCOPs and the GDL seemed attractive for the purposes of our work. However, thus far, neither GDL-based algorithms, nor DCOPs have been capable of meeting the complex interaction requirement. Furthermore, the effectiveness of these coordination algorithms when deployed in real agents to solve a real world problem has only been evaluated in simulation. Hence, their actual capability to meet the previously defined requirements when deployed in the real world still needs to be verified. Thus, we argued that a new body of research needs to be produced. In our case, this focused on two main challenges. First, the performance of these algorithms needs to be verified in practice. By addressing this challenge, we could start to confirm what they are actually capable of doing when confronted with the complexity of the real world. Second, a new theory needs to be defined to represent and solve problems involving agents engaged in complex interactions. By addressing this challenge, we could then derive new frameworks and algorithms to solve problems which have only been partially addressed by current research, but which are crucial to bridge the gap between the capabilities of existing algorithms and those that are required to solve complex real world problems. To achieve this, however, the first step is necessarily a theoretical one, since we first need to derive these algorithms and study their properties in a simple and controlled environment, before actually deploying them in the real world.

Against this background, the work presented in this thesis was divided into two main branches. In Chapter 3, we presented our practical contributions. These consist of a case study on the deployment of the max-sum algorithm, an approximate algorithm based on the GDL, on a system inspired by first responders at the scene of a disaster requesting imagery collection tasks of some of the most relevant areas to a team of UAVs. These agents then coordinate to complete the largest number of tasks. In chapters 4, 5, 6 and 7 we presented our theoretical contributions. These consist of a new class of problems and algorithms to address coordination problems involving complex agents' interactions (Chapter 4) and three examples of such interactions, namely multi-objective problems (Chapter 5), problems involving uncertainty and risk awareness (Chapter 6) and problems involving online learning (Chapter 7).

In more detail, the key idea of the system presented in Chapter 3 is to use the max-sum algorithm to allow the UAVs to provide first responders with accurate situational awareness in real time. As argued in Chapter 2, max-sum has been applied to a variety of problems related to situational awareness. However, a systematic framework that unifies all this literature into a principled methodology to guide a developer to the successful deployment of max-sum was still missing. To this end, we proposed such a methodology and, we described the way in which the algorithm satisfies the requirement of *generality* in the context of disaster response.

Now, in the system that we presented, the first responders use a personal digital assistant (a PDA) to request and receive such situational awareness in the form of a streaming video that the UAVs capture while hovering above a specific area. Within this system, the max-sum algorithm is fully decentralised between the different UAVs and PDAs. For this reason, it meets the requirement of *scalability* since new agents, either UAVs or PDAs, can be added or removed without affecting the system. It also meets the requirement of *robustness* since the performance of the system remains unaffected by the failure of either a UAV or a PDA. Finally, it meets the requirement of *resource awareness* because the computation required by max-sum is distributed efficiently between the available sources of computation (the UAVs and the PDAs).

The resulting system is evaluated both in simulation and in the real world. The former is used to confirm empirically the accuracy of the system. As a consequence, despite max-sum being an approximate algorithm, and not meeting the requirement of *performance guarantees*, we were still able to show that it yields a good performance nonetheless. Real world tests were then used to ascertain whether the algorithm satisfies one of the two key objectives of this thesis. More specifically, we ran these tests to illustrate the performance of max-sum in the real world. In particular, we deployed the algorithm on two hexacopters UAVs and tested it in a number of complex coordination scenarios. As indicated by these tests, both the system and max-sum responded well when confronted with the complexity and the unpredictability of the real world. In so doing, we were able to satisfy our first objective and demonstrate the potential of max-sum to coordinate real UAVs.

However, as previously stated, the max-sum and all GDL algorithms in general, are not yet capable of satisfying the complex interactions requirement in its entirety. For this reason, we dedicated the remaining chapters of this work, to discuss a new framework to represent such problems where the agents make complex joint decisions.

More specifically, the key reason why the current problem frameworks, including DCOPs, cannot represent such joint decisions lies in the way they measure the quality of these decisions. For instance, we discussed, in Chapter 2, how DCOPs measure the quality of each possible joint decision of multiple agents by using a real valued constraint function. However, such real valued functions are not capable of encompassing problems where the agents interact over complex decisions. This is because these are typically defined over multiple, heterogeneous parameters. Thus to represent such decisions, it is necessary to strike a balance between these dimensions. To deal with this, we proposed the framework of partially ordered distributed constraint optimisation problems (PO-DCOPs) in Chapter 4. We defined the framework as follows:

Partially ordered constraint optimisation problems (PO-DCOPs): A PO-DCOP generalises the constraint functions of a canonical DCOP into partially ordered vector

functions. Each vector function is then defined over the different parameters that define a complex interaction. In so doing, the use of such partial ordering allows us to generalise over the total order imposed by real valued functions. Consequently, we can represent and measure complex interactions that involve trading off between the above mentioned dimensions.

By introducing such PO-DCOPs we were then able to produce a novel framework capable of satisfying the *complex interactions* requirement. In addition, the PO-DCOP framework also satisfies the *generality* requirement. Indeed, no restrictions are made on the type of agents or the decisions that these agents are making. For this reason, in Chapters 5, 6 and 7 we proposed three novel frameworks that specialised PO-DCOPs to encompass three types of complex interactions. More specifically, we proposed the three following frameworks:

- **Multi-objective distributed constraint optimisation problems** (MO-DCOPs): This framework specialises PO-DCOPs to represent problems where the agents make decisions involving multiple objectives. We chose this type of interaction, since many real world coordination problems involve the simultaneous optimisation of multiple, possibly conflicting, objectives. In particular in disaster response, UAVs deployed to complete imagery collection tasks might need to coordinate over objectives such as maximising the number of completed tasks and minimising the passing through dangerous areas (e.g. areas containing buildings on fire or radiation). To trade off between these objectives, we introduced MO-DCOPs which re-define the constraint functions of a PO-DCOP as multiple objective functions. As a consequence, the solutions of these problems are partially ordered according to the Pareto dominance relation.

- **Risk-aware distributed constraint optimisation problems** (RA-DCOPs): This framework specialises PO-DCOPs to represent problems where the agents make decisions whose outcome is uncertain. This uncertainty implies that the value of the agents' joint decisions is defined according to a probability density function (pdf). Hence, in this setting, the agents need to carefully weight each of their decisions to avoid making poorly valued ones. To do so, they need to take into account the collective risk (i.e. the potential that one possible collective decision can lead to an undesirable outcome) associated with such decisions. We chose to focus on this type of interaction since uncertainty is endemic within many real world coordination problems. Considering again the disaster response setting, when UAVs are deployed to explore the scene of a disaster in search of casualties, they might not know with precision the number of casualties in the scene. However, this number is the key criteria to consider when selecting which area to explore next. Hence, the agents need to be able to trade off between the

risk of exploring areas with a very large, but highly uncertain, potential number of casualties (i.e. being risk seeking) and the risk of exploring areas with a lower, but certain, number of casualties (i.e. being risk averse). To achieve this, we defined RA-DCOPs which re-define the outcome of each possible assignment of a PO-DCOP's constraint function as a random variable. To determine the optimal solutions of these problems, a partial order relation was derived to rank the joint decisions' pdfs based on their risk.

- **Multi-arm bandit distributed constraint optimisation problems** (MAB-DCOPs): This framework specialises PO-DCOPs to represent problems where the agents make decisions whose outcome is uncertain and impossible to determine before the agents' deployment. This implies that the agents have to learn the outcome of their decisions online. We chose this type of interactions, since uncertainty and lack of prior information are characteristics pertaining to many real world settings. For instance, nothing might be known about the positions of the casualties before the UAVs are actually deployed. Hence, the UAVs might not know what the actual outcome of their decisions is going to be. For instance, they might assume that a certain area contains a high number of life-rafts, and thus, that exploring it might be highly rewarding. However, in reality the area might contain a limited number of casualties and, consequently, it is not the best area to explore. In these settings, the agents need to be able to trade off between making decisions that might reduce the uncertainty over the outcomes (*exploration*) and making decisions that might maximise their team performance (*exploitation*). To achieve this, we defined MAB-DCOPs which re-defines the constraint functions of a PO-DCOP as a MAB. To determine the optimal solution of these problems, we then imposed a partial order relation based on the upper confidence bound which ranked these solutions to achieve optimal regret (Chapter 2).

These three classes of problems, together with the general PO-DCOP framework, exhibit many of the same qualities as DCOPs. Most importantly, they satisfy the requirement of *generality*, since they can be used for multiple domains. However, by modelling the partially ordered constraint functions as vectors, some additional complexity is introduced. This is to be expected, since these new problems try to represent coordination problems while considering more real world details than canonical DCOPs.

In light of this, in Chapter 4, we proposed a new class of algorithms for solving PO-DCOPs by exploiting the qualities of the GDL. Here, the key idea was to follow the procedure used for canonical DCOP algorithms, which consists of exploiting the algebraic structure of DCOPs' constraint functions (i.e. the fact that they are defined over a *commutative semi-ring*), to obtain two operators, $\max_{\succ}$ and $\otimes$, which can be used to instantiate the GDL's message passing algorithms (Algorithms 2.1 and 2.2) to solve

PO-DCOPs. The key insight was to instantiate algorithms that proceed over three phases: (i) use graphical techniques, such as spanning or junction tree algorithms, to transform the problem so that it can be optimally solved by the GDL; (ii) use the GDL message passing algorithms to optimally solve the problem output by phase 1 and (iii) select a consistent solution by using value propagation. In so doing, given the properties of the GDL, we can instantiate optimal (DPOP, Action GDL) and bounded approximate (bounded max-sum) algorithms to solve PO-DCOPs. In so doing, all these algorithms meet three requirements that are fundamental for our work. First, they meet the *generality* requirement since they can solve general PO-DCOPs. Second, they satisfy the *quality guarantee* requirement since they are either optimal or can provide a bound on the quality of the solutions that they can find. Third, they satisfy the *robustness* requirement, since they are decentralised and so can cope with the failure of one or more components.

Next, we moved on to study some of the key properties of PO-DCOP algorithms to understand how they position themselves with respect to the requirements of our work. More specifically, we studied some of their key theoretical properties in Chapter 4 and empirically analysed their performance in Chapters 5, 6 and 7. In this context, our empirical analysis was focused on evaluating some of the fundamental properties of the algorithms. Specifically, we instantiated three of these algorithms to solve various instances of MO-DCOPs, RA-DCOPs and MAB-DCOPs respectively:

- **B-MOMS, a bounded approximate algorithms to solve MO-DCOPs**: We instantiated bounded multi-objective max-sum (B-MOMS) which solves MO-DCOPs while providing a bound on the quality of the recovered solutions. Next, we ran experiments on several MO-DCOP instances to verify the runtime performance of the algorithm, the approximation ratio and the quality of the solutions that it can produce. The results showed that B-MOMS was capable of providing good approximate solutions in a reasonable runtime, even for extremely large and complex problems. Hence, these results showed that B-MOMS satisfies both the requirements of *scalability* and *resource awareness*.

- **Action NDC-GDL, an optimal algorithm to solve RA-DCOPs**: We instantiated Action NDC-GDL, based on the Action GDL algorithm, which can solve RA-DCOPs while providing optimal solutions. Next, we ran experiments on several RA-DCOP instances to analyse their performance in terms of both computation and communication, most importantly by showing that using canonical DCOP techniques to solve these problems can lead to significant suboptimal performance. Our results show that solving RA-DCOPs optimally for problems involving up to 50 agents incurs a significant cost in terms of both computation and computation. However, our results further demonstrate a significant performance loss while using canonical DCOP techniques to solve RA-DCOPs. Hence,

our techniques fully satisfy the *scalability* requirement but only partially satisfy the *resource awareness* one. However, sacrificing such resources is necessary to produce effective solutions.

- **HEIST, an optimal algorithm to solve MAB-DCOPs**: We instantiate HEIST which can solve MAB-DCOPs by providing optimal regret on the quality of the solutions. Next, we ran experiments, on problem instances up to 50 variables, to ascertain the practical performance of HEIST in terms of solution quality, as well as communication and computation overhead. Our results showed that the algorithm was capable of achieving a reasonable trade off between the amount of computation and communication necessary to recover the solutions and their quality. Hence, these results demonstrated that the algorithm satisfies the *scalability* aspect and that it positions itself well with respect to the resource awareness requirement defined in Chapter 1.

When taken together, these experiments emphasized two key traits of our algorithms. First, bounded approximate algorithms perform well in the sense that they meet all the requirements defined in Chapter 1. Second, optimal algorithms meet all the requirements except resource awareness, which is only partially satisfied since solving PO-DCOPs optimally requires a bigger effort in terms of both communication and computation. This effort is further demonstrated in Chapter 4 where we analysed some of the key theoretical properties of PO-DCOPs algorithms. This analysis emphasized how both the size of the messages and the computation required to process them are exponential, not only in the number of variables but also in the number of parameters defining the PO-DCOPs' constraint vector functions.

8.2 Future Work

As we have seen in the previous section, the research presented in this thesis constitutes a significant step forward towards deploying cooperating agents in the real world. However, it is still a *first* step towards this aim. Hence, several challenges remain open and need to be addressed to push forward the work presented here.

In general, the algorithms that we derived to solve coordination problems involving complex interactions position themselves well with respect to our design requirements. However, the computation and the communication that these algorithms require can still be prohibitive for some real world problems. We argued, in the previous section, that this is to be expected because providing quality guarantees (either optimal or bounded approximate solutions) necessarily involves a large effort. Nonetheless, reducing this load is the key general challenge to address to bring this research closer to be deployed

to coordinate agents in the real world. In additon, by so doing, we would also be able to improve the way our algorithms behave with respect to the resource awareness requirement.

In light of this, we propose three key lines of research that naturally follow on from the current work. All of them share the same objective; that of reducing the amount of computation and communication required to compute the GDL messages. In particular, the idea is to define new techniques that are capable of reducing the size and amount of computation required to compute these messages. These techniques differ, however, in the way they try to achieve this objective:

- **Techniques that optimally preserve the quality guarantees**: The key aim of this research is to define new techniques that preserve the ability of our algorithms to provide quality guarantees, but improve these algorithms by using approaches to reduce the amount of computation required to compute GDL messages. To achieve this, a good starting point would be to study effective search space techniques such as branch and bound and branch and cut (see (Dechter, 2003)) to see whether they could be used to solve PO-DCOPs. Alternatively, there exist a variety of techniques that have been used to reduce the message size and the computation required by DPOP, which is a well known optimal GDL algorithm (see (Petcu, 2007) for more details). Thus, by using the same procedures, we could perhaps adapt these techniques to our settings and use them to solve PO-DCOPs.

- **Techniques that approximate and bound the quality guarantees**: The key aim of this research is to define new techniques to reduce the amount of communication required by the algorithm. The idea is to prun the edges of the factor graph that are likely to yield more communication. To achieve this, one possibility could be to generalise the approach described in (Macarthur et al., 2010) to the partially ordered case. To achieve this, we could define a new weighting approach where the weight of each edge trades off the edge's importance and its communication overhead. Thus, by pruning these edges, it would be possible to reduce the computation required by the algorithm by eliminating those edges that are likely to become bottlenecks.

- **Techniques that sacrifice the quality guarantees**: The key aim of this line of work is to define an approximate algorithm that can solve PO-DCOPs with very similar properties as the canonical max-sum algorithm. The reason for this choice was stated in Chapter 3, and is that max-sum is a very effective algorithm for coordinating agents in complex real world domains. The disadvantage in doing this is that max-sum does not provide any guarantee on the quality of the solutions that it recovers. However, strong empirical evidence, including the work presented in this thesis, has shown that it is capable of providing effective solutions nonetheless.

Unfortunately, deriving such an approximate GDL algorithm is extremely challenging when dealing with PO-DCOPs which might consist of several non comparable solutions. More specifically, aggregating these solutions to compute the GDL messages is currently impossible in loopy graphs. However, this problem can be avoided by considering that it is not necessary to consider all such solutions to compute an approximate solution. In fact, most of the equivalent solutions could be pruned, thus drastically reducing the amount of computation and communication required by the algorithm. However, to achieve this, it would be necessary to sacrifice the ability of the algorithm to provide quality guarantees. In this sense, the key challenge in defining these new techniques would be to show that they are capable of achieving a good compromise between the amount of computation and communication that they cut and the quality of the solution that they yield. A good starting point to derive these techniques would be studying pruning techniques such as alpha-beta or reduced error pruning used for data mining or search algorithms (see (Russel and Norvig, 1995) for a good introduction on these techniques) and see whether they can be applied to PO-DCOPs. In particular, the idea would be to define a set of techniques or, more generally, of criteria that can be used to cut those partial solutions that are less likely to yield an optimal one.

By deriving either of these techniques, we would be able to obtain extremely efficient algorithms that could be applied to several real world problems, even those involving agents with limited computational capacities. Achieving this is also another future challenge of this work. Indeed, we still need to illustrate the performance of our algorithms on more complex and dynamic coordination problems such as those presented in Chapter 2. To date, we have only tested our algorithms on various instantiations of the graph colouring problem, which is a well known and acknowledged test problem for coordination algorithms, but which still lacks some of the key challenges related to real world problems (see Chapter 5). In particular, it does not address dynamism or limited communication, which are often present in real world problems. In this sense, the problems discussed in Chapter 2 would constitute an excellent starting point to push forward the analysis of our algorithms. For instance, we could model the search and track problem as a MO-DCOP and instantiate either optimal or bounded approximate algorithms to solve them. We could then model the target tracking problem as a RA-DCOP. This would allow us to incorporate the uncertainty related to the UAVs' sensors in the decision making process. Finally, we could represent the SLAM problem as a MAB-DCOP where the agents have to learn the importance of their decisions (i.e. which area to explore next) depending on the number of casualties that they would find within each area. Moreover, these are not the only problems to which our algorithms can be applied. For this reason, it would be extremely interesting to see how these algorithms perform on domains such as the energy domain or the job scheduling domain which contains several features making them different from the situational awareness problems presented in

Chapter 2. In so doing, we would be able to gather sufficient knowledge and experience to again put our algorithms on real agents, as was done in Chapter 3, and demonstrate, finally, that they are the state-of-the-art for coordination for agents deployed in the real world.

Bibliography

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

M. Alighanbari and J. P. How. Decentralised Task Assignment for Unmanned Aerial Vehicles. In *Proceedings of the Fourty Fourth IEEE Conference on Decision and Control (CDC)*, pages 5668–5673, 2005. Sevilla, Spain.

M. Alighanbari and J. P. How. Robust Decentralised Task assignment for Cooperative Unmanned Aerial Vehicles. In *Proceedings of the AIAA Guidance Navigation and Control Conference (AIAA)*, pages 1–16, 2006. Keystone, USA.

Christopher Amato. *Increasing Scalability in Algorithms for Centralized and Decentralized Partially Observable Markov Decision Processes: Efficient Decision-Making and Coordination in Uncertain Environments.* PhD thesis, University of Massachusetts Amherst, 2010.

A. Antoniades, H. J. Kim, and S. Sastry. Pursuit-Evasion Strategies for Teams of Multiple Agents with Incomplete Information. In *Proceedings of the Fourty Second IEEE Conference on Decision and Control (CDC)*, pages 756–761, 2003. Maui, USA.

J. Atlas and K. Decker. Coordination for Uncertain Outcomes using Distributed Neighbor Exchange. *Proceedings of the Ninth International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, pages 1047–1054, 2010. Toronto, Canada.

P. Auer, N. Cesa-Bianchi, and P. Fischer. Finite–Time Analysis of the Multi-Armed Bandit Problem. *Machine Learning*, 47:235–256, 2002.

D. S. Bernstein, S. Zilberstein, and N. Immerman. The Complexity of Decentralised Control of Markov Decision Processes. In *Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 32–37, 2000. Stanford, USA.

B. Bethke, M. Valenti, and J.P. How. Uav Task Assignment. *IEEE Robotics & Automation Magazine*, 15(1):39–44, 2008.

F. Bourgault, T. Furukawa, and H. F. Durrant-Whyte. Coordinated Decentralized Search for a Lost Target in a Bayesian World. In *Proceedings of the IEEE International Conference on Intelligent Robots and Systems*, pages 48–53, 2003. Las Vegas, USA.

F. Bourgault, T. Furukawa, and H. F. Durrant-Whyte. Decentralized Bayesian Negotiation for Cooperative Search. In *Proceedings of the IEEE International Conference on Intelligent Robots and Systems*, pages 2681– 2686, 2004. Sendai, Japan.

E. Bowring, M. Tambe, and M. Yokoo. Multiply-Constrained Distributed Constraint Optimization. In *Proceedings of the Fith International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 1413–1420, 2006. ISBN 1-59593-303-4. Hakodate, Japan.

R. D. Braun, H. S. Wright, M. A. Croom, J. S. Levine, and D. A. Spencer. The Mars Airplane: A Credible Science Platform. *IEEE Aerospace Conference Big Sky MT*, pages 1–13, 2004.

M. Bryson and S. Sukkarieh. Building a Robust implementation of Bearing-only Inertial SLAM for a UAV. *Journal of Field Robotics*, 24(1-2):113–143, 2007.

F. Caballero, L. Merino, J. Ferruz, and A. Ollero. Improving Vision-Based Planar Motion Estimation for Unmanned Aerial Vehicles through Online Mosaicing. In *Proceedings 2006 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2860 – 2865, 2006. Orlando, USA.

E. F. Camacho and C. Bordons. *Model Predictive Control*. Springer, 2003.

A. Chapman, R. A. Micillo, R. Kota, and N. R. Jennings. Decentralised Dynamic Task Allocation: A Practical Game Theoretic Approach. In *Proceedings of the Eighth International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 915–922, 2009. Budapest, Hungary.

A. Chapman, A. Rogers, N. R. Jennings, and D. Leslie. A Unifying Framework for Iterative Approximate Best Response Algorithms for Distributed Constraint Optimisation Problems. *The Knowledge Engineering Review*, 26(4):411–444, 2011.

R. H. Chitale. *Probability and Queueing Theory*. Technical Publications, 2008.

D. T. Cole. *A Cooperative Unmanned Aircraft System Architecture for Information-Theoretic Search and Track*. PhD thesis, University Of Sydney, 2009.

R. Dechter. *Constraint Processing*. Morgan Kaufmann, 2003.

M.B. Dias, R. Zlot, N. Kalra, and A. Stentz. Market-Based Multirobot Coordination: A Survey and Analysis. *Proceedings of the IEEE*, 94:1257 – 1270, 2006.

P. Doshi and P. J. Gmytrasiewicz. Monte Carlo Sampling Methods for Approximating Interactive POMDPs. *Journal of Artificial Intelligence Research (JAIR)*, 34(1):297–337, 2009. ISSN 1076-9757.

E. W. Drew and J. Elston. Target Assignment for Integrated Search and Tracking by Active Robot Networks. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 2354–2359, 2008. Pasadena, USA.

H. M. Edwards. *Riemann's Zeta Function*. 2001.

E. Even-Dar, S. Mannor, and Y. Mansour. PAC Bounds for Multi-Armed Bandits and Markov Decision Processes. *Proceedings of the Fifteenth Conference on Learning Theory (COLT)*, pages 255–270, 2002.

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 Seventh International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, pages 639–646, 2008. Estoril, Portugal.

S. Fitzpatrick and L. Meertens. Distributed Coordination through Anarchic Optimization. In *Distributed Sensor Networks*, pages 257–295. Kluwer Academic Publishers, 2003.

E. W. Frew, D. A. Lawrence, and S. Morris. Coordinated Standoff Tracking of Moving Targets using Lyapunov Guidance Vector Fields. *Journal of Guidance, Control and Dynamics*, 31:290306, 2008.

B. J. Frey and D. Dueck. Clustering by Passing Messages between Data Points. *Science*, 315:972–976, 2007.

B. J. Frey, F. R. Kschischang, H. A. Loeliger, and N. Wiberg. Factor Graphs and Algorithms. In *Proceedings of the Allerton Conference on Communication Control and Computing*, volume 35, pages 666–680, 1997. Urbana, USA.

T. Furukawa, F. Bourgault, B. Lavis, and H. F. Durrant-Whyte. Recursive Bayesian Search and Tracking using Coordinated UAVs for Lost Targets. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, pages 2521–2526, 2006. Orlando, USA.

T. Furukawa, H. Durrant Whyte, and B. Lavis. The Element-Based Method - Theory and its Application to Bayesian Search and Tracking. In *Proceedings of the International Conference on Intelligent Robots and Systems (IROS)*, pages 2807–2812, 2007. Orlando, USA.

R. G. Gallager, P. A. Humblet, and P. M. Spira. A Distributed Algorithm for Minimum-Weight Spanning Trees. In *ACM Transactions on Programming Languages and Systems*, volume 5, pages 66–77, 1983.

C. E. Garcia, D. M. Prett, and M. Morari. Model Predictive Control: Theory and Practice: A Survey. *Automatica*, 25:335 – 348, 1989.

C. Geyer. Active Target Search from UAVs in Urban Environment. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 2366–2371, 2008. Pasadena, USA.

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

C. Guestrin, D. Koller, R. Parr, and S. Venkataraman. Efficient Solution Algorithms for Factored MDPs. *Journal of Artificial Intelligence Research (JAIR)*, 19:399–468, 2003.

M. Ham and G. Agha. Market-based Coordination Strategies for Physical Multi-Agent Systems. *SIGBED Review*, 5:23:1–23:2, 2008. ISSN 1551-3688.

J. P. Hespanha, H. J. Kim, and S. Sastry. Multiple-Agent Probabilistic Pursuit-Evasion Games. In *Proceedings of the Thirty Eighth IEEE Conference on Decision and Control (CDC)*, volume 3, pages 2432 – 2437, 1999. Phoenix, USA.

G. M. Hoffmann, S. L. Waslander, and C. J. Tomlin. Distributed Cooperative Search using Information-Theoretic Costs for Particle Filters with Quadrotor Applications. In *Proceedings of the AIAA Conference and Exhibit on Guidance, Navigation and Control*, pages 21–24, 2006. Keystone, USA.

G. Hollinger, S. Singh, J.Djugash, and A. Kehagias. Efficient Multi-Robot Search for a Moving Target. *The International Journal of Robotic Research*, 28:201–219, 2009.

J. P. How, C. Fraser, K. C. Kulling, L. F. Bertuccelli, O. Toupet, L. Brunet, A. Bachrach, and N. Roy. Increasing Autonomy of UAVs. *IEEE Robotics & Automation Magazine*, 16:43–51, 2009.

M. Jain, M. Taylor, M. Tambe, and M. Yokoo. DCOPs Meet the Real World: Exploring Unknown Reward Matrices with Applications to Mobile Sensor Networks. In *Proceedings of the Twenty-First International Joint Conference on Artificial Intelligence (IJCAI)*, pages 181–186, 2009. Pasadena, USA.

N. R. Jennings. An Agent-based Approach for Building Complex Software Systems. *Communications of the ACM*, 44:35–41, 2001.

Y. Jin, Y. Liao, M. Polycarpou, and A. Minai. Balancing Search and Target Response in Cooperative UAV Teams. In *Proceedings of the 43rd IEEE Conference on Decision and Control (CDC)*, pages 2923–2928, 2004. Nassau, Bahamas.

C. Kiekintveld, Z. Yin, A. Kumar, and M. Tambe. Asynchronous Algorithms for Approximate Distributed Constraint Optimization with Quality Bounds. In *Proceedings of the Ninth International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 133–140, 2010. ISBN 978-0-9826571-1-9. Toronto, Canada.

J. H. Kim and S. Sukkarieh. Airborne Simultaneous Localisation and Map-Building. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 406–411, 2003. Taipei, Taiwan.

D. Kingston, R. W. Beard, and R. S. Holt. Decentralized Perimeter Surveillance Using a Team of UAVs. *IEEE Transactions on Robotics*, 24:1394–1404, 2008.

F. R. Kschischang, B. J. Frey, and H. A. Loeliger. Factor Graphs and the Sum-Product Algorithm. *IEEE Transactions on Information Theory*, 47:498–519, 2001.

A. Kumar, B. Faltings, and A. Petcu. Distributed Constraint Optimisation with Structured Resource Constraints. In *Proceedings of the Eigth International Conference on Autonomous Agents and Multi-Agents Systems*, pages 923–930, 2009. Budapest, Hungary.

T. L. Lai and H. Robbins. Asymptotically Efficient Adaptive Allocation Rules. *Advances in Applied Mathematics*, 6(1):4–22, 1985.

T. Léauté and B. Faltings. E[DPOP]: Distributed Constraint Optimization under Stochastic Uncertainty using Collaborative Sampling. In *Proceedings of the IJCAI 2009 Distributed Constraint Reasoning Workshop (DCR)*, pages 87–101, 2009. Pasadena, USA.

T. Léauté and B. Faltings. Distributed Constraint Optimization under Stochastic Uncertainty. In *Proceedings of the Conference for the Association for the Advancement of Artificial Intelligence (AAAI)*, pages 68–73, 2011. San Francisco, USA.

V. Lesser, C. L. Ortiz, and M. Tambe. *Distributed Sensor Networks, A Multiagent Perspective*. Kluwer Academic Publishers, 2003.

H. Levy. *Stochastic Dominance: Investment Decision Making with Uncertainty*. Springer, 2006.

M. L. Littman. A Tutorial on Partially Observable Markov Decision Processes. *Journal of Mathematical Psychology*, 53:119 – 125, 2009.

D. Ma and A. Zhang. Coretracking: An Efficient Approach to Clustering Moving Targets and Tracking Clusters. In *Proceedings of the IEEE Radar Conference*, page 117122, 2004. Philadelphia, USA.

K. Macarthur, R. Stranders, S. D. Ramchurn, and N. R. Jennings. A Distributed Anytime Algorithm for Dynamic Task Allocation in Multi-Agent Systems. In *Proceedings of the Conference for the Advancement of Artificial Intelligence (AAAI)*, pages 356–362, 2011. San Francisco, USA.

K. S. Macarthur, A. Farinelli, S. D. Ramchurn, and N. R. Jennings. Efficient, Superstabilizing Decentralised Optimisation for Dynamic Task Allocation Environments. In *Proceedings of the AAMAS 2010 Workshop on Optimisation in Multi-Agent Systems (OptMas)*, 2010. Toronto, Canada.

D. J. C. MacKay. Good Error-Correcting Codes based on very Sparse Matrices. *IEEE Transactions on Information Theory*, 45:399 –431, 1999. ISSN 0018-9448.

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 *Proceedings of the Fourth International Conference of Autonomous Agents and Multi-Agent Systems (AAMAS)*, pages 127–146, 2005. New York, USA.

R. Marinescu. Exploiting Problem Decomposition in Multi-Objective Constraint Optimization. In *Proceedings of the Fifteenth International Conference on Principles and Practice of Constraint Programming (CP)*, pages 592–607. 2009. ISBN 3-642-04243-0, 978-3-642-04243-0. Lisbon, Portugal.

R. T. Marler and J. S. Arora. Survey of Multi-Objective Optimization Methods for Engineering. *Structural and Multidisciplinary Optimization*, 26:369–395, 2004.

C. McDiarmid. On the Method of Bounded Differences. In *Surveys in Combinatorics*, number 141 in London Mathematical Society Lecture Note Series, pages 148–188. Cambridge University Press, 1989.

M. Mezard, G. Parisi, and R. Zecchina. Analytic and Algorithmic Solution of Random Satisfiability Problems. *Science*, 297:812–815, 2002.

S. Miller, S. D. Ramchurn, and A. Rogers. Optimal Decentralised Dispatch of Embedded Generation in the Smart Grid. In *Proceedings of the Eleventh International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, 2012. Valencia, Spain.

P. J. Modi, W. M. Shen, M. Tambe, and M. Yokoo. Adopt: Asynchronous Distributed Constraint Optimization with Quality Guarantees. *Artificial Intelligence*, 161:149–180, 2005.

K. P. Murphy, Y. Weiss, and M. I. Jordan. Loopy Belief Propagation for Approximate Inference: An Empirical Study. In *Proceedings of the Conference for the Uncertainty in Artificial Intelligence (UAI)*, pages 467–475, 1999.

J. Ousingsawat and M. E. Campbell. On-line Estimation and Path Planning for Multiple Vehicles in an Uncertain Environment. *International Journal of Robust and Nonlinear Control*, 14(8):741–766, 2004.

V. Pareto. *Manual of Political Economy*. Macmillan, 1906. Translated by Ann S. Schwier and Alfred N. Page, 1972.

J. P. Pearce, R. T. Maheswaran, and M. Tambe. How Local is that Optimum? k-Optimality for DCOP. In *Proceedings of the Fourth International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, pages 1303–1304, 2005. ISBN 1-59593-093-0. New York, USA.

A. Petcu. *A Class of Algorithms for Distributed Constraint Optimization*. PhD thesis, Swiss Federal Institute of Technology (EPFL), Lausanne (Switzerland), 2007.

A. Petcu and B. Faltings. DPOP: A Scalable Method for Multi-Agent Constraint Optimization. *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence (IJCAI)*, pages 266 – 271, 2005. Edinburgh, Scotland.

A. Rogers, D. D. Corkill, and N. R. Jennings. Agent Technologies for Sensor Networks. *IEEE Intelligent Systems*, 24:13–17, 2009.

A. Rogers, A. Farinelli, R. Stranders, and N. R. Jennings. Bounded Approximate Decentralised Coordination via the Max-Sum Algorithm. *Artificial Intelligence*, 175: 730–759, 2011.

E. Rollón. *Multi-Objective Optimization in Graphical Models*. PhD thesis, Universitat Politecnicá de Catalunya, 2008.

E. Rollón and J. Larrosa. Bucket Elimination for Multi-Objective Optimization Problems. *Journal of Heuristics*, 12:307–328, 2006.

N. Roy. *Finding Approximate POMDP Solutions Through Belief Compression*. PhD thesis, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, September 2003.

S. J. Russel and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice-Hall, 1995.

Z. Sarris. Survey of UAV Applications in Civil Markets. In *Proceedings of the Ninth IEEE Mediterranean Conference on Control and Automation*, pages 1 – 11, 2001. Dubrovnik, Croatia.

P. Scerri, E. Liao, Y. Xu, M. Lewis, J. Lai, and K. Sycara. Coordinating Very Large Groups of Wide Area Search Munitions. *Theory and Algorithms for Cooperative Systems*, pages 1 – 31, 2005.

E. Sommerlade and I. Reid. Information Theoretic Active Scene Exploration. In *Proceedings of the IEEE Computer Vision and Pattern Recognition (CVPR)*, pages 1–7, 2008. Anchorage, USA.

S. Stolle and R. Rysdyk. Flight Path Following Guidance for Unmanned Air Vehicles with pan-tilt Camera for Target observation (dasc 03). In *Proceedings of the 22nd Digital Avionics Systems Conference (DASC)*, volume 2, page 8, 2003. Indianapolis, USA.

R. Stranders. *Decentralised Coordination of Information Gathering Agents*. PhD thesis, University of Southampton, 2010.

R. Stranders, F. M. Delle Fave, A. Rogers, and N. R. Jennings. A Decentralised Coordination Algorithm for Mobile Sensors. In *Proceedings of the Conference for the Association of the Advancement for Artificial Intelligence (AAAI)*, pages 874–880, 2010. Atlanta, USA.

R. Stranders, A. Farinelli, A. Rogers, and N. R. Jennings. Decentralised Coordination of Mobile Sensors Using the Max Sum Algorithm. In *Proceedings of the Twenty First International Joint Conference on Artificial Intelligence (IJCAI)*, pages 299–304, 2009. Pasadena, USA.

S. Sukkarieh and H. F. Durrant-Whyte. Towards the Development of Simultaneous Localisation and Map Building for an Unmanned Air Vehicle. In *Proceeding of the International Conference of Field and Service Robotics*, pages 193–200, 2001. Helsinki, Finland.

E. A. Sultanik, P. J. Modi, and W. C. Regli. On Modeling Multi-Agent Task Scheduling as a Distributed Constraint Optimisation Problem. In *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1531–1536, 2007. Hydebarad, India.

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

A. Symington, S. Waharte, S. Justin Julier, and N. Trigoni. Probabilistic Target Detection by Camera-Equipped UAVs. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 4076–4081, 2010. Anchorage, USA.

E-G. Talbi, S. Mostaghim, T. Okabe, H. Ishibuchi, G. Rudolph, and C. Coello. Parallel Approaches for Multi-Objective Optimization. In Jrgen Branke, Kalyanmoy Deb, Kaisa Miettinen, and Roman Slowinski, editors, *Multiobjective Optimization*, volume 5252 of *Lecture Notes in Computer Science*, pages 349–372. Springer Berlin / Heidelberg, 2008.

M. Taylor, M. Jain, Y. Jin, M. Yokoo, and M. Tambe. When Should There be a "Me" in "Team"?: Distributed Multi-Agent Optimization Under Uncertainty. *Proceedings of the Ninth International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, pages 109–116, 2010. Toronto, Canada.

S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. Intelligent Robotics and Autonomous Agents. The MIT Press, 2005.

S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, J. Diebel, P. Fong, J. Gale, M. Halpenny, and G. Hoffmann. Stanley: The Robot that won the DARPA Grand Challenge. *Journal of Field Robotics*, 23:1–43, 2006.

J. Tisdale, R. Allison, Z. Kim, D. T ornqvist, and J. Karl Hedrick. A Multiple UAV System for Vision-based Search and Localisation. In *Proceedings of the American Control Conference (ACC)*, pages 1985–1990, 2008. Seattle, USA.

UAS Roadmap. U.s. Army Unmanned Aircraft Systems Roadmap (2010 - 2030). Technical report, Office of the Secretary of the Defence, 2010.

K. Valavanis and K.P. Valavanis. *Advances in Unmanned Aerial Vehicles: State of the Art and the Road to Autonomy*. Springer Verlag, 2007.

D. A. Van Veldhuizen and G.B. Lamont. Multi-Objective Evolutionary Algorithm Research: A History and Analysis. Technical report, Department of Electrical and Computer Engineering Graduate School of Engineering, Air Force Institute of Technology, 1998.

J. Vermorel and M. Mohri. Multi-Armed Bandit Algorithms and Empirical Evaluation. *Proceedings of the Sixteenth European Conference on Artificial Intelligence (ECML)*, pages 437–448, 2005. Porto, Portugal.

R. Vidal, S. Rashid, C. Sharp, S. Jin, and S. Sastry. Pursuit-Evasion Games with Unmanned Ground and Aerial Vehicles. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 2948–2955, 2001. Seoul, Korea.

M. Vinyals, J. A. Rodriguez-Aguilar, and J. Cerquides. Constructing a unifying Theory of Dynamic Programming DCOP Algorithms via the Generalised Distributive Law. *Autonomous Agents and Multi-agent Systems (JAAMAS)*, 22:439–464, 2011.

M. Vinyals, J.A. Rodriguez-Aguilar, and J. Cerquides. Egalitarian Utilities Divide and Coordinate: solving DCOPs by Agreement. In *Proceedings of the Ninth International Conference of Autonomous Agents and Multi-Agent Systems (AAMAS)*, pages 150–156, 2010. Toronto, Canada.

P. Vytelingum, S. D. Ramchurn, T. D. Voice, A. Rogers, and N. R. Jennings. Trading Agents for the Smart Electricity Grid. In *Proceedings of the Ninth International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 897–904, 2010. Toronto, Canada.

S. Waharte, A. Symington, and N. Trigoni. Probabilistic Search with Agile UAVs. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2010. Anchorage, USA.

S. Waharte and N. Trigoni. Supporting Search and Rescue Operations with UAVs. In *Proceedings of the 2010 International Conference on Emerging Security Technologies (EST)*, pages 142–147, 2010. ISBN 978-0-7695-4175-4. Washington, USA.

G. Xu and Z. Zhang. *Epipolar Geometry in Stereo, Motion, and Object Recognition: A Unified Approach*. Springer, 1996.

Y. Yang, A. A. Minai, and M. M. Polycarpou. Evidential Map-Building Approaches for Multi-UAV Cooperative Search. In *Proceedings of the 2005 American Control Conference (ACC)*, pages 116–121, 2005. Portland, USA.

U. Zengin and A. Dogan. Real-Time Target Tracking for Autonomous UAVs in Adversarial Environments: A Gradient Search Algorithm. *IEEE Transactions on Robotics and Automation*, 23:294–307, 2007.