UNIVERSITY OF SOUTHAMPTON
Faculty of Physical and Applied Sciences
School of Electronics and Computer Science
Intelligence, Agents, Multimedia Group

Multi-Objective Decentralised Coordination for Teams of Robotic Agents

by Francesco M. Delle Fave

December 6, 2010

A mini-thesis submitted for transfer from MPhil to PhD

Supervisors: Professor Nicholas R. Jennings and Doctor Alex Rogers
Examiner: Professor Sandor M. Veres

UNIVERSITY OF SOUTHAMPTON

ABSTRACT

FACULTY OF PHYSICAL AND APPLIED SCIENCES
SCHOOL OF ELECTRONICS AND COMPUTER SCIENCE

A mini-thesis submitted for transfer from MPhil to PhD

by Francesco M. Delle Fave

This thesis introduces two novel coordination mechanisms for a team of multiple autonomous decision makers, represented as autonomous robotic agents. Such techniques aim to improve the capabilities of robotic agents, such as unmanned aerial or ground vehicles (UAVs and UGVs), when deployed in real world operations. In particular, the work reported in this thesis focuses on improving the decision making of teams of such robotic agents when deployed in an unknown, and dynamically changing, environment to perform search and rescue operations for lost targets. This problem is well known and studied within both academia and industry and coordination mechanisms for controlling such teams have been studied in both the robotics and the multi-agent systems communities.

Within this setting, our first contribution aims at solves a canonical target search problem, in which a team of UAVs is deployed in an environment to search for a lost target. Specifically, we present a novel decentralised coordination approach for teams of UAVs, based on the max-sum algorithm. In more detail, we represent each agent as a UAV, and study the applicability of the max-sum algorithm, a decentralised approximate message passing algorithm, to coordinate a team of multiple UAVs for target search. We benchmark our approach against three state-of-the-art approaches within a simulation environment. The results show that coordination with the max-sum algorithm out-performs a best response algorithm, which represents the state of the art in the coordination of UAVs for search, by up to 26%, an implicitly coordinated approach, where the coordination arises from the agents making decisions based on a common belief, by up to 34% and finally a non-coordinated approach by up to 68%. These results indicate that the max-sum algorithm has the potential to be applied in complex systems operating in dynamic environments.

We then move on to tackle coordination in which the team has more than one objective to achieve (e.g. maximise the covered space of the search area, whilst minimising the amount of energy consumed by each UAV). To achieve this shortcoming, we present, as our second contribution, an extension of the max-sum algorithm to compute bounded solutions for problems involving multiple objectives. More precisely, we develop the bounded multi-objective max-sum algorithm (B-MOMS), a novel decentralised coordination algorithm able to solve problems involving multiple objectives while providing guarantees on the solution it recovers. B-MOMS extends the standard max-sum algorithm to compute bounded approximate solutions to multi-objective decentralised constraint optimisation problems (MO-DCOPs). Moreover, we prove the optimality of B-MOMS in acyclic constraint graphs, and derive problem dependent bounds on its approximation ratio when these graphs contain cycles. Finally, we empirically evaluate its performance on a multi-objective extension of the canonical graph colouring problem. In so doing, we demonstrate that, for the settings we consider, the approximation ratio never exceeds 2, and is typically less than 1.5 for less-constrained graphs. Moreover, the runtime required by B-MOMS on the problem instances we considered never exceeds 30 minutes, even for maximally constrained graphs with one hundred agents.

Contents

List of Figures

List of Tables

Nomenclature

x_k^T	A variable representing the position of a target T at time step k
o_k^m	A matrix representing the value of the observation taken by agent m at time k
o^k	A matrix representing the fusion of the observations taken by the entire team of agents
P_k^T	A matrix representing the common belief shared by the agents over the target's position at time k
$J(\mathbf{u}, N)$	The control function thate the agents use to decide the next set of actions $\mathbf{u}$ given a receding horizon control of N time steps
$q_{j\rightarrow i}(x_j)$	A max-sum message sent by variable x_j to function U_i
$r_{i\rightarrow j}(x_j)$	A max-sum message sent by function U_i to variable x_j
U	A general global utility function (single objective case)
$\mathbf{U}$	A general vector of global utility functions (multi-objective case)
U_i	A general local utility function with $i \in [1, \dots, N]$ is the index of the specific constraint (single objective case)
$\mathbf{U}_i$	A general vector of local utility functions, with $i \in [1, \dots, N]$ (multi-objective case)
U_i^k	The local utility function corresponding to objective k (multi-objective case)
$\mathbf{U}_m$	The values of the solutions related to the pruned factor graph.
$\tilde{\mathbf{U}}$	The values of the solutions of the approximated problem
x_j	A discrete variable with $j \in [1, \dots, M]$
a_j	An assignment for variable x_j
$\mathbf{a}$	An assignment to all the variables
$\mathbf{a}^*$	The optimal assignment to the variables.
w_{ij}	A weight of the edge of a factor graph between function U_i and variable x_j

w_{ij}^k	A weight of an edge of the factor graph related to objective k.
$\mathbf{w}_{ij}$	A vector of weights w_{ij}^k.
$\mathbf{W}$	The sum of all the weights $\mathbf{w}_{ij}$ pruned from the factor graph
V	The utopia point.

Chapter 1

Introduction

Recent years have seen both academia and industry increase their interest in settings where groups of agents need to operate as a team to solve common goals. Such goals characterise the different aspects of the problem that the agents are solving and are commonly referred to as the different objectives of the problem. In many of these applications, the agents represent robotic platforms such as unmanned aerial or ground vehicles (UAVs and UGVs), and are deployed in an environment to gather information about some specific features, such as spatial phenomena (Stranders et al., 2009), or for mission involving different objectives such as searching and tracking a group of unknown targets (Furukawa et al., 2006). In both cases, the information collected by the different agents is then used to make decisions over which action each of them should take next.

Within these settings, research shows that the performance of the team is deeply influenced by the way in which the agents make these decisions about their actions, and that such performance decreases whenever they act independently from one another (Cole, 2009; Grocholsky, 2002; Lesser et al., 2003). Thus, a key challenge for this type of problem is to find ways in which the members of the team can coordinate their decision processes in order to increase the overall performance of the collective. Moreover, such decision processes have to take into account all the different objectives that might define the operation, such as searching for lost targets while simultaneously tracking the ones that have already been found. Indeed, operations such as gathering information in large environments with teams of robotic agents, are inherently characterised by multiple, possibly conflicting, objectives that need to be optimised at the same time (Cole, 2009; Inalhan et al., 2002). However, most current research lacks methodologies that enable these same agents to make decisions over multiple objectives. Moreover, the current state-of-the-art coordination approaches have typically been evaluated only on simple test-problems, such as graph-colouring (Farinelli et al., 2008; Chapman et al., 2011) or for coordinating teams of sensors (Stranders et al., 2009), and not in the rich coordination settings in which they are ultimately intended to be deployed. Addressing these two short comings is the focus of this thesis.

1.1 Motivations

The aim of this research is to develop effective coordination mechanisms for teams of agents representing robotic platforms such as UAVs or UGVs. In particular, we wish to improve upon current coordination techniques in order to enhance the deployment of these agents in real world domains where teams of agents operate in unknown and possibly hostile environments to acquire information.

Research literature related to this topic, usually adopts some canonical scenarios in order to motivate the importance of studying coordination for teams of such agents and to describe the various requirements, challenges and issues that arise when coordinating teams of robotic agents in real world domains. The focus of this thesis is on scenarios involving teams of UAVs tasked to search for lost targets in an unknown environment (Bourgault et al., 2003, 2004). However, we aim to develop general techniques, which are, therefore, applicable to other type of platform and to other type of settings as well.

Given this setting, the canonical scenario involves a large ship sinking whilst at sea and the ship's crew (and possibly the passengers if it is, for instance, a cruise ship) requesting help and abandoning the ship in multiple life-rafts. The task of the team of UAVs is then to support their human operators in order to effectively rescue all the passengers of the ship. The objectives of the team of UAVs are, then, twofold:

- To search for and find all the life-rafts and communicate this information to the rescue services.
- To track the position of all the different life-rafts that have been found, in order to enable the rescue services to reach and rescue them.

We decided to focus on robotic platforms such as UAVs, because they have been advocated as the best platform to use to search large outdoor environments like the one encountered in the scenario. This is principally due to their sensing range, larger than the one of vehicles restricted to the ground, which enables these vehicles to cover open areas more efficiently than vehicles restricted to the ground. Indeed, UAVs constitute a unique platform in terms of the information they can provide (Cole, 2009) and they have been used in many different types of mission including civil operations, such as border interdiction, wild fire suppression and communications relay (Sarris, 2001), and military operations including chemical cloud detection, meteorology missions and maritime operations for ship classification (Sarris, 2001; Kovacina et al., 2002).

Furthermore, this type of scenario introduces various challenges that must be taken into account in order to successfully complete the mission. Clearly, the rescue operation must be accomplished in the least amount of time since ensuring the safety of all the passengers of the ship is critical. In order to search and locate the different life-rafts each

UAV will take observations of its surroundings using sensors like fixed cameras. However, such measurements are likely to be inaccurate due to the precision of the sensors and to various external factors such as the weather conditions, the sea conditions and the time in which the operation is taking place. Furthermore, each of these UAVs has a limited amount of fuel, therefore efficiently handling the energy resources of each vehicle becomes a fundamental objective of the operation as well. The life-rafts will be drifting over the sea, therefore, their position will be constantly changing. Communication with the headquarters, with the human team, and between the different platforms might not always be possible, because the communication of each UAV is limited in range. Some of these UAVs might fail due to various component failures. Moreover, humans operators might not always be available or might be overloaded with other tasks. Finally, the large size of the area will require a large number of UAVs to be deployed.

In addition, the scenario depicts a canonical information gathering problem for the team of agents, namely that of search and track. In this problem a team of agents operates in an environment searching for some unknown targets. Once some of this targets are found, part of the team is tasked to track their positions, while the remaining members search the environment for the remaining targets, or for those targets whose position has been lost. Each agent is provided with a fixed camera sensor to constantly observe the environment in order to find the targets. Such a camera is used in a similar fashion to acquire information about the current location of an already identified target.

In this context, agents make decisions depending on the quality of the information provided by the observations that they have taken (Bourgault et al., 2003, 2004; Cole, 2009; Stranders et al., 2010). More specifically, every observation contains some information, thus each agent builds its own local belief about the state of the targets and uses this belief to make a decision. For instance, if a platform observes a specified area and does not find a target, it updates its beliefs and decides to search an unexplored area. In such a setting, the aim of the agents is then to optimally explore the environment in order to find the lost targets in the least amount of time, while simultaneously acquiring precise knowledge about the position of those targets that have already been found.

Moreover, as mentioned in the scenario, agents possess limited resources, therefore managing them in an efficient fashion becomes another key objective of the mission. In other words, the mission is characterised by three different objectives:

- The agents need to explore the search area in order to find all the different drifting life-rafts.
- The agents need to track the life-rafts which have already been found, in order to maintain precise information about their position.
- The agents need to effectively manage their limited energy resources.

Clearly, the principal objective of the team of agents is to rescue all the civilians in the least amount of time. To accomplish this task in an effective manner, however, it is necessary to provide each agent with a set of clear objectives that it is able to fulfil during the operation. For this reason, the operation is characterised as the former set of three sub-objectives, which are well known tasks for teams of robotic agents (Furukawa et al., 2006, 2007).

Now, these objectives are also incommensurate because the search task, the tracking task and the resource management process are three separate objectives that need to be completed by the same team of agents. Furthermore, these objectives are conflicting and cannot be solved independently. To understand this, we consider the following example:

Example 1.1. *Consider a scenario where two agents (u_1 and u_2) are tasked to search for the life-rafts while simultaneously monitoring their energy resources. The first area is unexplored, therefore, it has a high value in terms of the search objective. Unfortunately, this area is situated in a zone characterised by a strong wind, therefore exploring it will require a large amount of energy. Conversely, the second area has previously been explored by another agent of the team, but a significant amount of time has passed since then. It is then reasonable to assume that some life-rafts might have reached it during this time interval, therefore the area needs to be explored again. Furthermore, this second area is in a low-wind zone, therefore its exploration will consume a reasonably low amount of energy. Finally, u_1 has more energy left than u_2.*

Given Example 1.1, assuming that each agent makes its decision independently and that each agent owns the same amount of information then, the task of each agent will consist of deciding which of the two different areas each agent it is going to explore next. Given this setting, if the agents consider only the search objective in their decision making process, they will then both explore the first area because it is unexplored. Therefore, they will consume a large amount of energy resources, which will affect their mission time (specifically u_2's mission time), and therefore their overall team performance. Conversely, if the agents consider both the objectives, that is, not only searching but handling its energy resources as well, they will then both decide to explore the second area. The reason for this is that the second area, while having already been explored, requires less energy. More specifically, this area is less important then the first area in terms of the search objective, because it has already been explored. However, by choosing the second one, the performance of the team over this objective will not be affected greatly, because the area needs to be explored again. Moreover, considering the objective related to managing energy resources, choosing the second area is the best possible decision, because the agents will consume less energy and their mission time will be longer. Thus, choosing the second area is the best possible decision as it will maximise the overall performance of the team. In a similar fashion, it is easy to show how tracking is conflicting with the other two objectives.

Moreover, Example 1.1 illustrates a scenario where each agent makes decisions independently, therefore, considering only their local belief about the targets and/or about energy resources. However, by adopting this decision making process, the agents might end up exploring the same regions, therefore, they would redundantly cover the search area. Moreover, each agent would consider only its own energy resources to make decisions, therefore, an agent might end up exploring extremely expensive areas that could have been explored more effectively by some other members of the team. Hence, this type of approach would lead to a sub-optimal consumption of the energy resources.

Thus, the overall performance of the team is likely be poor in terms of the three objectives of the problem and thus, the time to complete the mission would increase. The key point here, is that independent decision making does not exploit the advantages that considering the behaviour of other members of the team might bring. Considering again the example. If the agents were to make their decisions together, they could decide to explore both the areas. Specifically, u_2, having less energy, would explore the second area, while u_1 would explore the first one. For this reason, and in order to achieve better performance, the agents need to make decisions together, considering the behaviour of all the members of the team. Specifically, they need to consider how these behaviours are correlated and how the established global behaviour affects the team performance. In short, they need to coordinate their decisions.

To achieve this, the agents need to share all the information necessary for their decision process. This exchange is usually accomplished by sending (and receiving) messages regarding each agent's beliefs and possible decisions (Grocholsky, 2002). More specifically, agents share two types of messages, *observations* messages and *coordination* messages. The former contains the information about the observations taken by an agent (e.g. information about the areas that have already been searched, or information about the position of the target that is currently being tracked). This information is used to fuse the beliefs of every member of the team into a common global belief that each agent uses to make decisions (Grocholsky, 2002). The latter contains information about the impact that a team decision would have on the operation (e.g. by following a specific trajectory, or by searching a specific area). In this context, the importance of this information is directly related to the team performance and is commonly defined as a global utility over the team decision. Such global utility is calculated considering the importance of the contribution of every member, which is defined as its local utility. Each of these local utilities measures how each agent's decisions affect the global utility (Cole, 2009).

Against this background, the aim of this research is to develop coordination algorithms that improve the performance of teams of robotic agents performing information gathering tasks such as searching and tracking targets such as life-rafts. More specifically, we wish to develop general techniques that coordinate such a team of robotic agents, which can be either UAVs or UGVs, when deployed in a dynamic and uncertain environment, to achieve multiple and incommensurable objectives, such as gathering different types of information and managing energy resources. Now from the above discussion, let us distill a set of functional and non-functional requirements that will shape the design of our coordination algorithms. In particular, we can identify three, fundamental, functional requirements:

Multi-objective: The agents make decisions over multiple objectives. Such objectives may be incommensurate, and they cannot be optimised independently because they are conflicting. Thus, improving one objective may decrease the value of another. Our coordination approaches need, therefore, to be able to address these issues in an effective way.

Quality Guarantees: Many aspects of an operation are uncertain. This affects the outcome of the coordination techniques. For instance, sensor measurements and communication are likely to be imprecise and thus the agents will not be able to ascertain with precision which areas have already being explored or the amount of energy consumed by other agents. In these settings, coordination mechanisms cannot guarantee to achieve an optimal decision process. Thus, a fundamental requirement for our coordination approaches is to be able to provide guarantees on the quality of the decisions that they compute.

Timeliness: The decision process needs to be carried out in *real-time*. In other words, the agents need to make their decisions while simultaneously gathering data about the environment and flying towards their next destination. Moreover, during the interval of time, that each agent uses in order to make a decision, the environment may change considerably due to its dynamism. Therefore, the decision time needs to be as short as possible. A coordination approach should, therefore, allow the agents to rapidly deliberate and converge to the best possible team decision given the common beliefs and the available time.

We can further identify several non-functional properties that will be considered to develop our coordination approach:

Adaptiveness: As depicted in the scenario, information about the state of the targets is uncertain. This is mostly due to the poor information known at the beginning of the mission and to the imprecision related to the measurements provided by the sensors. Moreover, the life-rafts are drifting, therefore their state changes over time. For these reasons, the team of agents should be able to continuously adapt its decisions based on the knowledge that it continuously gathers about the different objectives. In more detail, the coordination mechanism should allow the agents to continuously modify their decisions, by taking into account all the information that they have collected.

Robustness: As depicted in the scenario, communication between members of the team is limited, therefore a coordination approach should be able to provide good team performance even when communication between the different agents is limited. Moreover, should one of the agents fail as well, the operation of the remaining platforms should be affected as little as possible. More generally, our coordination algorithm needs the ability to sustain a good level of performance in the face of unexpected events, and to degrade gracefully as component failures occur or bandwidth diminishes.

Autonomy: As depicted in the scenario, a human operator might not be available, thus every platform should be able to control its own actions without being governed by an external system. This implies that an agent should be capable of making its own decisions, based on the global common belief that it has been able to compile from its own observations and communication with other members of the team. In this context, autonomy is related to robustness since autonomy means that there is no central system that remotely controls the platforms. Having such a system would make a robotic agent, such as a UAV or a UGV, highly vulnerable when the command link is severed. Moreover, autonomous platforms do not require human attention which is often a scarce resource.

Scalability: This property refers to two key aspects of our approaches. On one hand, it should be possible to scale up the size of the team of agents, if the problem requires more agents to be solved properly. This property refers, then, to the ability of the coordination algorithm to handle the interactions with a growing number of neighbouring platforms. On the other hand, scalability also refers to the number of objectives that the coordination approach should be able to handle. In this thesis, we will focus on scenarios involving at most one hundred agents and three objectives[1].

In order to meet these requirements, pre-computing search plans for each agent offline (i.e. before the operation starts), is not advisable for two fundamental reasons. First, automated planning cannot address the dynamism associated with tracking the life-rafts, which will clearly affect the performance of the team. Second, such approaches cannot handle in a timely fashion the higher level of uncertainty that characterises many features of the mission, such as the size of the searching area, the measurements of the sensors and the positions of the targets, because such parameters are not known before the mission effectively starts. Furthermore, a centralised approach is not advisable either, since the agents should not be dependent on a single central controller which might fail. Thus, the use of a decentralised online approach to coordination is needed. More specifically, decentralisation properly addresses the issues of robustness to failures of individual agents, since it removes the dependency on a central system, hence removing

[1]We chose such number because, to the best of our knowledge, it is the highest number used in relevant literature to test the scalability of this type of approaches (Rollón, 2008).

a potential single point of failure (Lesser et al., 2003) and finally it allows the team to increase its scale, because the computational effort is split between the different agents (Lesser et al., 2003; Scerri et al., 2005). The use of an online approach further allows the team to adapt to the uncertainty and dynamism related to the different objectives and to the environment.

Against this background, current research has addressed only a subset of these requirements. In particular, as we will see in more detail in Chapter 2, three different types of decentralised coordination mechanisms for teams of robotic agents have been defined, depending on the type of information that they share (Cole, 2009):

Non-coordinated approaches where the platforms behave independently, making autonomous individual decisions.

Implicitly coordinated approaches where the platforms share their observations, thus building a common world view before making autonomous individual decisions. Hence, the coordination mechanism is implicit, because the agents do not take into account the behaviour of the other members of the team.

Explicitly coordinated approaches where the agents share their observations and then jointly decide what to do next. Hence, in this case, each agent now explicitly considers the behaviour of the other members of the team.

For information gathering tasks, explicit coordination approaches are usually preferable since they have been demonstrated to achieve better performance than implicit and non-coordinated approaches (Bourgault et al., 2004; Cole, 2009). Such approaches also address some of the non-functional requirements that we defined earlier; specifically, they are autonomous, adaptive and able to scale to an arbitrary number of platforms (Cole, 2009). However, to date, most coordination mechanisms fall short in meeting the robustness requirement. In particular, the explicit coordination mechanisms developed thus far are often not robust against component and communication failure, since the performance of the team is severely affected whenever one of the agents fail, or whenever communication is not fully available (Cole, 2009). Most importantly, existing approaches do not meet the functional requirements that we defined. The timeliness requirement is not met since the agents often converge extremely slowly to an effective team decision (Bourgault et al., 2004). Moreover, they do not address problems involving multiple objectives, nor can they provide guarantees on the quality of the solutions that they have found.

The research presented in this thesis addresses these specific shortcomings. In the next section, we outline our contributions towards this direction.

1.2 Research Contributions

The aim of this research is to improve the coordination capabilities of teams of robotic agents, such as UAVs or UGVs in order to enhance their deployment in real world domains. Specifically, we wish to contribute to the field by developing an integrated set of techniques which fulfil the requirements defined in the previous section, therefore allowing such robotic platforms to coordinate in a timely, robust, fashion over multiple objectives and to provide quality guarantees over their decisions. Moreover, we aim to develop techniques as general as possible, and thus applicable to a wide variety of domains and to different platforms.

To accomplish this, we first focus on a canonical setting, similar to the one presented in the previous section, in which a team of UAVs is deployed into an unknown environment to search for a lost target. Within these domain, we start our study by analysing the current literature on robotics and multi-agent systems. In particular, the literature on robotics lacks two fundamental aspects related to the coordination of teams of robotic agents. First, to date, only simple coordination mechanisms, such as non-coordinated or implicitly coordinated (Bourgault et al., 2003; Cole, 2009), have been applied, for coordinating robotic agents in information gathering problems, which are not guaranteed to achieve the adequate level of performance given the requirements of the problem. Second, no general purpose coordination mechanisms exist, that can address problems involving multiple objectives. While coordination mechanisms have been developed to address specific domains such as searching and tracking groups of targets (Cole, 2009; Furukawa et al., 2006), they cannot easily be extended to address problems involving more than these two objectives, such as, for instance, considering, as well, the management of energy resources, as mentioned in the scenario. Moreover, existing approaches do not scale up to a higher number of objectives, which is a fundamental requirement for addressing real world domains.

Conversely, within the multi-agent community, recent literature on decentralised coordination has shown that online decentralised coordination mechanisms, based upon the max-sum algorithm (Farinelli et al., 2008), are effective solution techniques to a number of benchmark problems, including graph-colouring (Farinelli et al., 2008) and the coordination of mobile sensors that are constrained to move within a graph (Stranders et al., 2009, 2010). In such problems, the max-sum algorithm has been shown to outperform other explicit coordination approaches such as best response, and to be robust to communication and component failure (Farinelli et al., 2008). Moreover, the max-sum algorithm has been recently extended in order to provide quality guarantees over the solutions that it computes, and has again been shown to provide solutions extremely close to the optimal for graph colouring problems (Farinelli et al., 2009).

All the above features make max-sum an attractive technique to use for our research. However, despite having been extended to compute bounded solutions, the max-sum algorithm cannot handle multiple objectives. Moreover, the algorithm has not been applied, yet, to the problem of coordinating specific platforms such as UAVs, that move in an unconstrained continuous space.

Therefore, we present in this thesis, our contributions in order to address these shortcomings. In more detail, we first address, the problem of studying how to apply the max-sum algorithm to a canonical information gathering problem, namely target search (Cole, 2009; Bourgault et al., 2003, 2004). Within this framework, we present a study of how the max-sum algorithm can be applied to the coordination of UAVs tasked to search this continuous space. We benchmark our approach against three state-of-the-art approaches (a best response, an implicitly coordinated and a non coordinated approach). Our empirical results show that the max-sum approach out-performs all the aforementioned approaches, and therefore constitutes an effective approach to use for coordinating teams of UAVs.

Second, we address the problem of generalising the max-sum algorithm in order to compute bounded solutions to multi-objective problems. More specifically, we define and characterise the bounded multi-objective max-sum algorithm (B-MOMS), the first general decentralised algorithm to compute bounded solutions over problems involving multiple objectives. In these settings, we prove a number of theoretical properties of our algorithm and empirically evaluate its performance on a multi-objective extension of the canonical graph colouring problem and demonstrate that B-MOMS satisfies the requirements defined in Section 1.1.

In more detail our contributions are the following:

Decentralised Coordination for Teams of UAVs (Chapter 3): For the first time, we apply the max-sum algorithm to the challenging task of coordinating a team of UAVs to search for a target in a continuous space. By doing this, we introduce a novel coordination technique to the field of robotic search, and we extend the max-sum algorithm beyond the simpler coordination problems to which it has been applied to date (Farinelli et al., 2008; Stranders et al., 2009). Moreover, we benchmark our algorithm against three existing approaches that have been proposed for coordinating UAVs for search and show that it out-performs an explicitly coordinated approach based on a best response algorithm (Bourgault et al., 2004) by up to 26%, an implicitly implicitly coordinated approach by up to 34% and finally a non-coordinated approach by up to 68%.

Bounded Multi-Objective Decentralised Coordination (Chapter 4): We introduce the multi-objective distributed constraint optimisation (MO-DCOP) problem. This is the first formalisation of a multi-objective coordination problem, which generalises the well known DCOP framework (Modi et al., 2005) to the multi-objective setting. We then derive B-MOMS, the first bounded decentralised algorithm for solving multi-objective optimisation problems. The operation of B-MOMS consists of three phases:

- The first phase builds upon the bounded max-sum algorithm (Rogers et al., 2011) and extends the approach to provide quality guarantees by computing a cycle-free graph over the constraint graph representing the MO-DCOP.
- The second phase extends the max-sum algorithm to the multi-objective domain in order to optimally solve the cycle-free graph resulting from the first phase. To achieve this, we generalise the key mathematical operators required by max-sum to optimally solve the multi-objective problem encoded by the cycle-free graph.
- Since there may be multiple optimal[2] assignments to the cycle-free problem, the third and final phase enables agents to reach consensus on which global assignment to choose.

We further prove that B-MOMS is optimal in acyclic factor graphs, and derive problem dependent approximation bounds in general graphs that do contain cycles. Finally, 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. We demonstrate that the approximation ratio never exceeds 2, even for extremely constrained problems (i.e. fully connected graphs), and is less than 1.5 for graphs where constraints exists between 20% of all pairs of agents. Moreover, our 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.

The work up to now has led to a published paper to an international workshop and to one submission at an international conference:

1. F.M. Delle Fave, Z.Xu, A. Rogers, and N.R. Jennings. Decentralised coordination of unmanned aerial vehicles for target search using the max-sum algorithm. In *Proceedings of the AAMAS 2010 Workshop on Agents in Real-Time and Dynamic Environments*, 2010. Toronto, Canada.
2. F.M. Delle Fave, R. Stranders, A. Rogers, and N.R. Jennings. Bounded Decentralised Coordination over Multiple Objectives. *Submitted to the Tenth International Conference on Autonomous Agents and Multi-Agent Systems* (AAMAS 2011).

Now, it is important to note that the contributions mentioned above, solve all the requirements presented earlier, but do not address two more challenges that were depicted

[2]Note that, as we will see in Chapter 2, in the multi-objective domain, optimality is defined in terms of the set of *Pareto* optimal assignments of a problem.

in the scenario. Specifically, the scenario emphasises two challenges to address in order to promote the use of autonomous robotic platforms in real world domains:

- Coordination mechanisms need to address *uncertainty* over the global utility function that the agents are maximising. The scenario depicts only one source of uncertainty, the imprecision related to the sensor measurements. However, such uncertainty is *endemic* in real world domains and current state-of-the-art coordination mechanisms fall short in providing an effective countermeasure against it.
- Coordination mechanisms needs to be tested on field experiments over real UAVs. The scenario describes the team of UAVs as having been extensively tested on field experiments. However, to date, only trivial coordination approaches have been properly implemented in unmanned aircraft systems (UAS) (Cole, 2009).

Such challenges constitute the future work that we are going to address for the remaining part of this PhD. We will detail such challenges and how we intend to deal with them in Chapter 5.

1.3 Report Outline

The remainder of the report is organised as follows:

- In Chapter 2, we discuss related work. We initially analyse the benefits of using unmanned aerial vehicles for information gathering tasks and describe the information gathering tasks they are typically applied to. We then introduce multi objective optimisation, by characterising the general problem, its well known solution concept—Pareto optimality—- and by outlining the different algorithms constituting the state-of-the art. Finally, we discuss decentralised coordination, present the max-sum algorithm and its corresponding extension to compute bounded solutions.
- In Chapter 3, we present our first contribution; that is, a decentralised coordination algorithm for teams of UAVs for target-search. We initially introduce the problem, and then characterise our solution technique. We conclude the chapter presenting the empirical evaluations that were run in order to test the performance of our algorithm.
- In Chapter 4, we present our second contribution; that is, the bounded multi-objective max-sum algorithm. We first introduce the MO-DCOP problem, the general multi-objective problem that our algorithm solves. We then characterise in detail the three phases of our algorithm, prove its optimality over acyclic graphs

and the approximation ratio that it calculates. Finally, we present an extensive empirical evaluation of our approach against an optimal brute-force algorithm.

- In Chapter 5 we present our conclusions and determine which steps need to be undertaken in future work. Specifically we talk about the two challenges presented in Section 1.2 and how we intend to address them.

Chapter 2

Related Work

We review, in this chapter, the literature relevant to our work. We focus primarily on the literature pertaining to our specific problem. Thus, we focus on one specific information gathering problem, the target search problem and rule out those coordination approaches that do not satisfy the requirements discussed in Chapter 1.

In more detail, in Section 2.1 we provide an overview on current unmanned aerial vehicles. In Section 2.2, we review the target search problem and its extension that incorporates tracking the targets that have been found. In Section 2.3, we describe the decentralised coordination approaches existing in literature and evaluate their effectiveness in our framework. We focus specifically on the max-sum algorithm for coordination, because it constitutes the key milestone over which we build our coordination approaches. In Section 2.4, we discuss multi-objective optimisation. First, we define two well known solution concepts for multi-objective problems. Second, we analyse the current state-of-the-art approaches, by pointing out the reasons why they are not suitable for the coordination of teams of UAVs for target search. Finally, Section 2.5 summarises the chapter.

2.1 Unmanned Aerial Vehicles

Unmanned aerial vehicles (UAVs) have been principally used for military applications in recent years (UAS Roadmap, 2005). Indeed, such vehicles were created, developed and studied mainly for operations, such as reconnaissance surveillance and target acquisition (RTSA), maritime operations (naval fire support and ship classification) and meteorology missions (Sarris, 2001). However, recent years have also seen the number of civil operations involving UAVs increase greatly. Many successful operations such as search and rescue, communications relay and disaster and emergency management have been carried out by teams of UAVs, and such operations are expected to increase over the coming years (Sarris, 2001; Valavanis and Valavanis, 2007).

FIGURE 2.1: the MQ1-Predator during an operation

Such a wide variety of operations has led industry to develop a large range of different types of UAVs. Moreover, in order to categorise such vehicles, different types of taxonomies have been proposed. The most important among these, classifies a UAV depending on the altitude that it can reach, its endurance and flying range (Sarris, 2001). To provide some examples, Figure 2.1 shows an example of a medium-altitude (up to 9000 m), long-endurance (over 200 km) UAV, the MQ-1 Predator, widely used for reconnaissance and combat missions in Iraq, Bosnia and Afghanistan by the United States Air Force.

Figure 2.2 shows a picture of the RQ4 Global Hawk, one of the largest UAV developed to date, 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 (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, 2005).

Now, all the UAVs presented so far are remotely controlled by human operators (Valavanis and Valavanis, 2007). This type of unmanned vehicle, also referred as a drone, currently constitutes the most commonly deployed type of UAV for real world operations. However, research on autonomous control has made great progress in recent

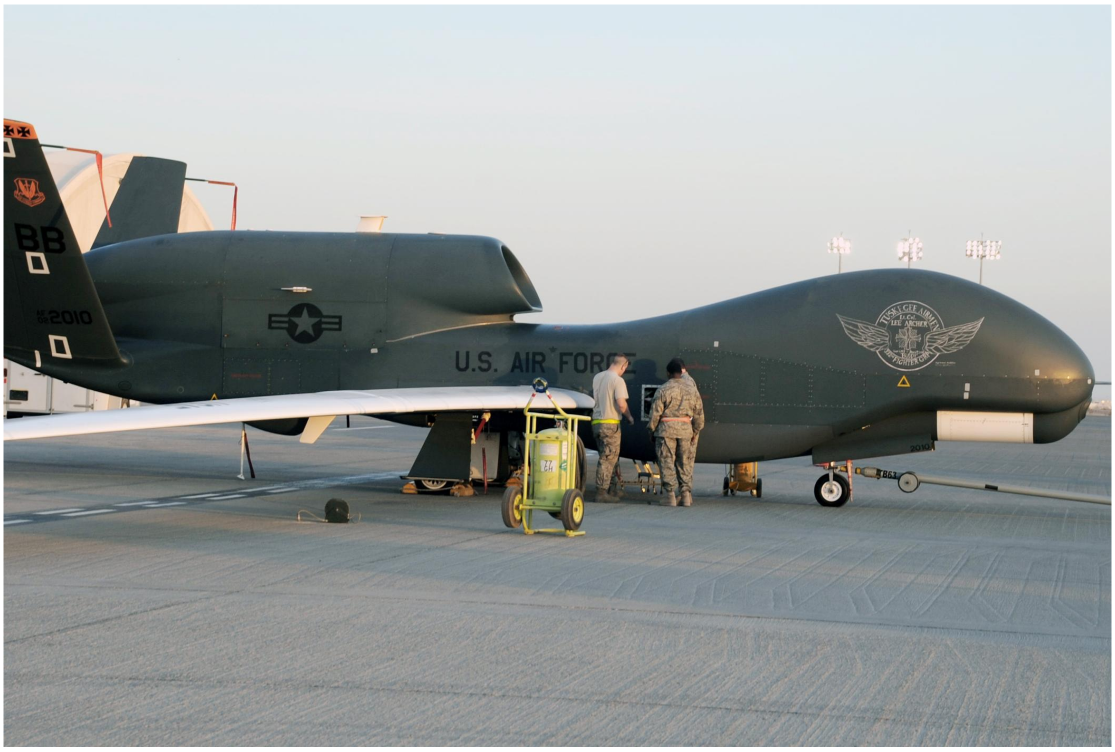


Figure 2.2: the RQ4 Global Hawk

years, and is predicted to increase further in the future (UAS Roadmap, 2005). Hence, industry is already investing money and effort into developing completely autonomous UAVs. An example is the Herti developed by BAE SYSTEMS and shown in figure 2.3. Herti is a recently developed vehicle, that has been successfully deployed in Afghanistan for trial missions[1].

Moreover, all the above-mentioned vehicles represent highly advanced technological assets, whose cost can reach tens of millions of American dollars (UAS Roadmap, 2005). For the above reason, to date, only a small number of these platforms have been deployed (Sarris, 2001). Other types of UAVs, commonly used for civil applications are presented in Figure 2.4. Such vehicles are known as micro-UAVs, albeit in recent years this definition has been attributed to smaller vehicles. These UAVs are used for a wide range of applications from surveillance of pipelines to scientific research. A wide variety of these vehicles currently exist, and they more commonly being used and developed by industry and academia, due to their lower costs (Sarris, 2001).

Now, the previous discussion outlined how unmanned aerial vehicles are commonly used for information gathering missions, such as surveillance or reconnaissance. As can be seen in Figures 2.1 and 2.3, they are provided with various sensing devices, such as gimballed or fixed cameras (Xu and Zhang, 1996), which can provide high quality in-

[1]For more detailed information, refer to the BAE Systems news releases web page, ref. 357/2007, 06 Nov. 2007 `http://www.baesystems.com/Newsroom/NewsReleases/2007/autoGen_107106174325.html`

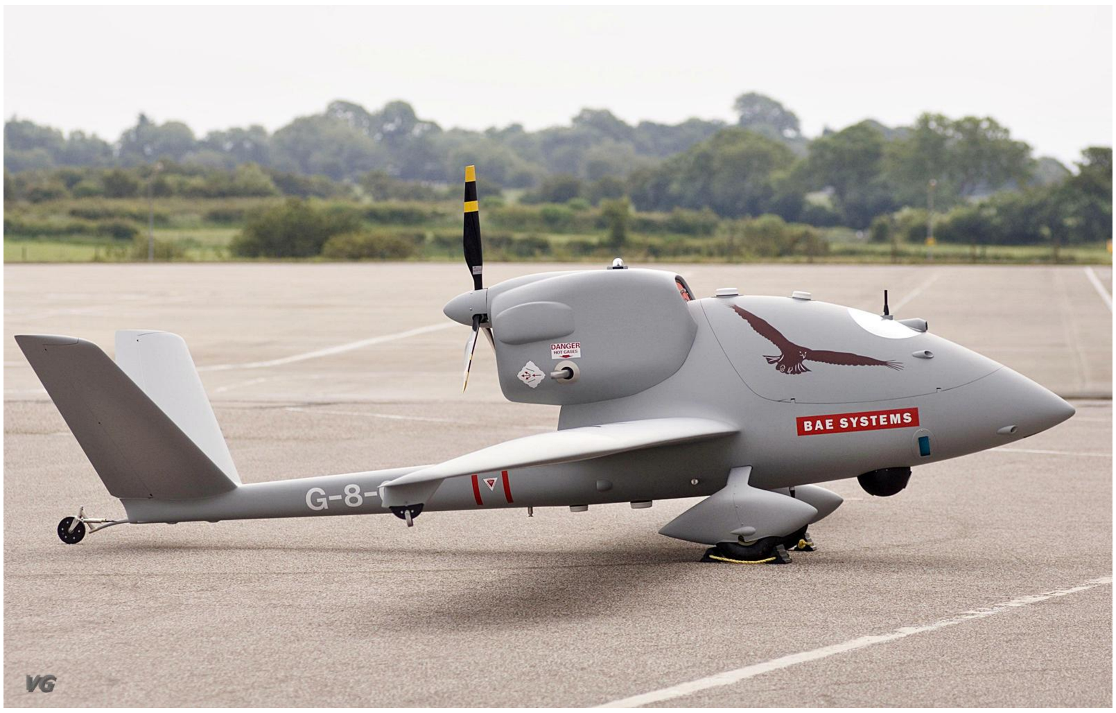


FIGURE 2.3: the BAE-Herti

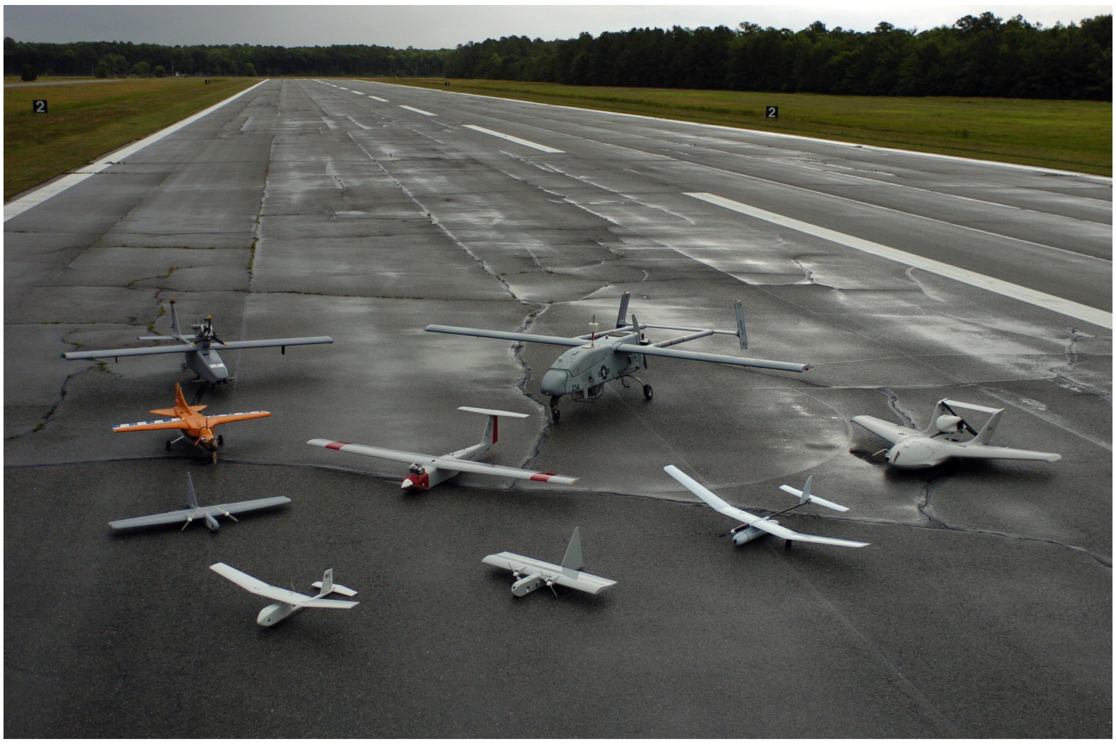


FIGURE 2.4: An illustration of different types of micro-UAVs

formation about any type of features within their flying range. Moreover, recent state-of-the-art research on autonomous control for teams of multiple platforms, being done by groups such as the Australian Centre for Field Robotics (ACFR) or the Stanford Artificial Intelligence Laboratory (SAIL), is finally starting to show significant results (Grocholsky, 2002; Cole, 2009), and it has been estimated that by 2030 teams of UAVs will be able to perform real missions with complete autonomy (UAS Roadmap, 2005). For these reasons, UAVs are commonly considered the state-of-the-art in terms of platforms to use for autonomous information gathering missions, such as the target search or target tracking (Cole, 2009).

For the purposes of our work, we will abstract from the specific features of the platform and will represent each UAV as an autonomous agent. This choice will provide a high level of flexibility and generality to our representation and is reasonably standard in the robotics research community (Bourgault et al., 2003, 2004; Cole, 2009). Moreover, since our aim is to develop general coordination techniques (i.e. applicable to a wide variety of platform and of coordination scenarios), the use of abstract agents allows us to not be tied to any specific platform.

2.2 The Target Search Problem

The target search problem belongs to the class of problems known as information gathering tasks (Cole, 2009). The canonical scenario is the same as the one presented in Chapter 1 and refers to the problem of searching an area in order to find one or more, lost targets (Bourgault et al., 2003). Here, searching refers to the task of detecting a target (i.e. having it within the sensing range of at least one vehicle). However, in many real scenarios, such as the one depicted in Chapter 1, the agent is also required to precisely localise the positions of the targets. Thus, the agents must also track the targets in order to maintain a precise estimate of their position (Furukawa et al., 2006). This latter problem, referred to as the search and track problem, extends target search by incorporating the problem of tracking the targets that have already been found.

In the remainder of this section, we discuss in detail literature on the canonical search problem in Section 2.2.1, and on the search and track problem in Section 2.2.2.

2.2.1 The Search Problem

The canonical description of the search problem involves a team of different UAVs that search an unknown environment in order to find some unknown targets. Bourgault et al. (2003) characterises the problem of searching for a single target as an information gathering task and use a well known grid-based approach to share a common belief about the possible position of the target between the different UAVs. This problem has been further extended in order to enrich the search scenario with features that would make it more detailed and, therefore, more applicable to real world domains.

Some of these features focus specifically on the type of environment or target considered. Examples include extending the search problem in order to address the search of groups of targets when their total number is unknown (Cole, 2009), considering more complex searching areas such as an urban environment (Geyer, 2008), or considering a hostile environment containing threats and/or obstacles (Vidal et al., 2001; Yang et al., 2005). Moreover, the search problem has been extended to the case of adversaries who are actively trying to avoid detection (Bopardikar et al., 2008). Further extensions focus on aspects related to the formalisation of the problem. Some of these abstract out the real world setting by representing the searching area as a graph (Hoffmann et al., 2006) or an occupancy grid (Hespanha et al., 1999; Vidal et al., 2001; Antoniades et al., 2003). Within these settings, the agents, representing the UAVs, move by jumping from one cell or node to another. Moreover, such approaches also often simplify the sensor model by having the agents simply observe the state of a subset of the cells or nodes (Hespanha et al., 1999; Vidal et al., 2001; Antoniades et al., 2003), indicating whether the target is there or not.

Conversely, other approaches adopt a more realistic representation in which the area of interest is represented as a continuous space where each platform moves using a continuous motion model based on speed and heading (Bourgault et al., 2003, 2004; Cole, 2009). Furthermore, every agent is provided with a sensor model that accurately simulates its actual sensor and the information about the environment that it provides. Examples of sensor models for information gathering tasks include the bearing only sensor model (Hoffmann et al., 2006), that measures the direction or bearing to the target, and the camera model (Bourgault et al., 2003, 2004) commonly used to represent the fixed camera (Xu and Zhang, 1996) that each UAV uses to observe the region of the area that it is exploring.

Now, all these different versions of the target search problem share a common, fundamental, feature. In all the scenarios, the different UAVs continuously observe the environment in order to spot the targets and maintain belief about their possible position. Such uncertain information is constantly updated with new observations taken by the different UAVs, and is used in order to decide which portion of the environment to explore next. Thus, modelling the information about the state of the target plays a fundamental role for solving the target search problem.

Specific literature on target search often characterises the uncertainty regarding the target's location by defining a probability density function (PDF) over the search area. The main task is then defined as minimising this uncertainty by observing areas where the level of uncertainty is high (Cole, 2009). This probability density function is often approximated by using a discrete grid where each cell represents a specific area of the search space and the cell value represents the probability of finding the target in this area (Bourgault et al., 2003, 2004; Cole, 2009). The precision of the model is strictly related to the resolution of the grid. A larger number of cells represents the area in more detail. However, increasing the resolution of the grid increases the computation required to store and maintain this map. For this reason a second approach, based on a particle filter, is often used (Hoffmann et al., 2006). In this second approach, the probability map is discretised into a set of particles, representing random samples of the PDF, and a Bayesian filter is used to estimate and update the value of these particles (Thrun et al., 2005). This approach results in a lower computational cost and a higher precision compared to the previous one (Hoffmann et al., 2006). However, particle filters depend on a large set of parameters which require a sophisticated tuning to function properly (Thrun et al., 2005). For this reason, research on target search has principally adopted the grid-based approach in order to represent knowledge.

The different agents use the above mentioned PDF in order to make a decision over which area to explore next. Specifically, such agents use an optimisation approach, where a global utility function, defined over the PDF, is optimised and the solutions are used in order to determine the next areas to explore. Some approaches define this utility function as the probability of detecting the target in a specific sub-region of the search space (Bourgault et al., 2003, 2004), while other approaches use an information metric, based on the Shannon entropy, to measure the reduction in uncertainty that results from the observation of a specific area (Cole, 2009).

In order to understand the challenges and the issues that arise when dealing with more realistic and complex domains, we focus initially on studying the canonical search problem. Indeed, many issues arise when dealing with a more realistic setting, such as communication limitations and uncertainty about the targets and the team members state. Within this setting, as we will show in detail in Chapter 3, we define a PDF to represent the uncertainty about the target's location and use a Bayesian grid approach to estimate it. We choose to use such an approach because it represents the state-of-the-art for fusing beliefs in a decentralised fashion within the search domain (Bourgault et al., 2003, 2004; Cole, 2009). We then represent the search area as a continuous space and we define a continuous motion model and a fixed camera model for each UAV. Finally, the use of a camera sensing model, allows us to represent uncertainty in a more expressive and complete fashion than using a simple discrete probability. A detailed explanation of how the camera model is used to take measurements of the state of the target, and how this is fused together in order to build a discretised representation of its position within the area of interest will be given in the following chapter.

2.2.2 The Search and Track Problem

In the search and track framework a team of UAVs is tasked with searching for multiple targets, while simultaneously tracking (i.e. localising) the ones that have already been found. The problem has been formalised by Furukawa et al. (2006) for the single target case[2] and subsequently extended to multiple targets in (Furukawa et al., 2007). Ideally, this scenario can be seen as a target assignment problem where the team of UAVs cooperate in order 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 the search and track have been mostly extensions to the ones used for the standard search problem. In particular, Furukawa et al. (2006), characterises the problem as an information gathering problem and presents a unified approach for solving it. The same representation of the belief over the target position is used for both search and track. A Bayesian grid is used and different types of observation updates are defined for the tracking task (i.e. when the target is detected) and for the search task (i.e. when the target is not detected). A similar approach is used by Tisdale et al. (2008) who further includes the probability of false detection (i.e. sensors are not perfect). In this same work, a comparison is made between particle and grid based approaches for approximating the PDF. In this work, grid based methods are shown to be as effective as particle filters for the search task, however, the higher degrees of accuracy required for estimating the target's positions requires the use of particle filters. A different approach is proposed by Furukawa et al. (2007), where, an element based method, using shape functions as a more computationally efficient means of representing the PDF is introduced and shown to be more effective than grid-based methods.

A task assignment formalisation of the search and track problem is introduced by Drew and Elston (2008). 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

[2]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 enough information has been gathered.

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

In this thesis, we present a general algorithm in Chapter 3 to address the canonical search problem (i.e. without considering tracking targets). However, we will present in Chapter 4 a general algorithm to address problems involving the simultaneous optimisation of multiple objectives, and will apply it to search and track for future work.

2.3 Decentralised Coordination Approaches

As mentioned in Chapter 1, we rule out from our study offline and centralised coordination approaches, because they do not meet the fundamental requirements to coordinate teams of UAVs. Decentralised coordination approaches have been studied in both the multi-agent system (MAS) and robotics communities. A well known classification of these approaches is based on the type of information shared by the different agents (Cole, 2009; Grocholsky, 2002). Specifically, decentralised coordination approaches can be divided into three levels:

- Non-coordinated approaches where all the agents behave autonomously and do not take into account the information gathered by the other members of the team.
- Implicitly coordinated approaches where the agents share their observations, thus building a common belief about the state of the target. This belief is then used to make individual decisions on what action to take next. Hence, the decisions of the other members of the team are not taken into account. This approach is also often referred as the coordinated approach in robotics literature (Grocholsky, 2002).
- Explicitly coordinated approaches where the agents share both observations and decisions about what actions to take next. In this case, the agents now explicitly take into account the behaviour of their teammates when making their own decision. This approach is also referred as the cooperative approach in robotics literature (Grocholsky, 2002).

All three types of approach share, however, a common feature. Indeed, in each of them, the agents constantly build and update a dynamic model of the target's position, which they then use in order to make a decision over which area to observe next.

In the remainder of this section, we discuss each of the coordination approaches presented above in the context of target search, and then present in detail the max-sum algorithm, the second fundamental contribution that we take from literature.

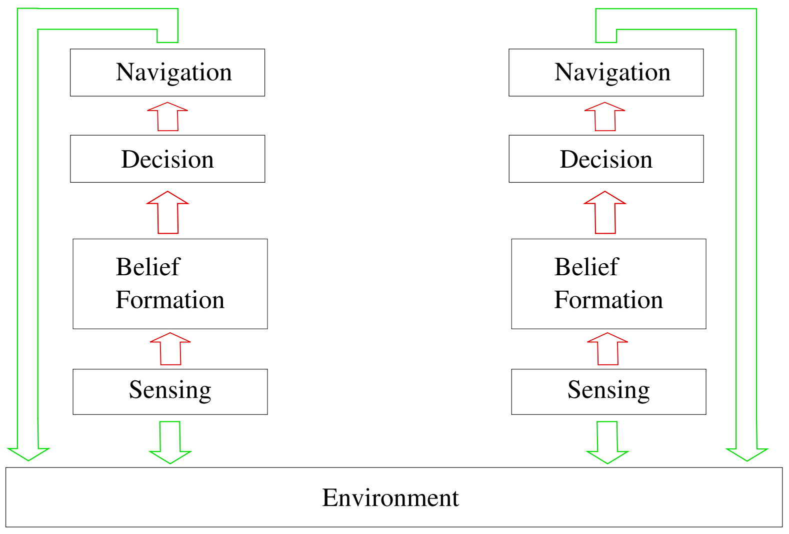


FIGURE 2.5: The architecture of a non-coordinated approach (Cole, 2009)

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. Indeed, as defined at the beginning of the section, non-coordinated approaches are those in which the agents do not share any type of information with one another. This means that each agent makes decisions relying only on the information that it has gathered itself. Figure 2.5 illustrates the canonical architecture for a non-coordinated approach, where, each agent makes its own decisions independently, without sharing any type of information with the other members of the team.

For this reason, greedy techniques are commonly considered as the best possible method to use in order to define non-coordinated approaches, where each agent decides to observe the best location, given only the local information that it has gathered about the position of the target (Vidal et al., 2001; Antoniades et al., 2003). In the target search domain, a local greedy algorithm has been defined for the discretised version of problem, where each agent aims for the location, among those in its neighbourhood, that will most reduce the uncertainty over the target's position (Hespanha et al., 1999; Vidal et al., 2001; Antoniades et al., 2003). A similar approach has been used in a more realistic setting, where each UAV locally decides which area to explore next, by greedily finding the control action that minimises the uncertainty over the area (Hoffmann et al., 2006).

Such uncertainty over the target's position is then used in order to predict its next position. To achieve this, approaches based on receding horizon control have been considered for solving information gathering problems (Bourgault et al., 2003, 2004; Cole, 2009). More specifically, such approaches provide an estimate of the position of the target by predicting its behaviour over a specific time horizon[3]. In this context myopic approaches

[3]Such approaches are also referred as Model Predictive Control in related literature (Camacho and

have been employed, where agents make decisions over a single-step time horizon, that is, over the immediate future, therefore, not taking into account the estimated model of the target. In so doing, each agent maximises the probability of detecting the target without considering its dynamism. For this reason, myopic techniques often lead to a poor performance in terms of the coverage of the search area and in terms of the time required to detect the target. Thus, longer time horizons have also been taken into account in order to achieve a better approximation (Hoffmann et al., 2006).

The main issue when dealing with a finite horizon is finding the reasonable threshold between its length (i.e. the quality of the estimate) and its feasibility. Many practical approaches require a time horizon that is impossible to calculate, since it involves computing more complex decisions, and therefore, finding a good overall joint decision requires a higher computational effort. For this reason, greedy techniques based on short time horizons are commonly considered the best approaches in order to compute feasible solutions for scenarios where agents do not coordinate (Hoffmann et al., 2006).

Now, the previous discussion describes independent behaviours for each agent. Crucially when a team of agents is deployed, each agent should be considering the behaviour of the other members in order to increase the overall performance. For this reason, non-coordinated approaches are not advisable when dealing with multiple agents. Nonetheless, such approaches constitute a reasonable lower bound for the performance of more sophisticated coordination algorithms. Thus, they have been widely used in coordination literature to benchmark the performance of implicit and explicit approaches (Bourgault et al., 2003, 2004; Hollinger et al., 2009; Cole, 2009), and will be used in this work to benchmark the coordination approach that we are going to present in Chapter 3.

2.3.2 Implicitly Coordinated Approaches

Implicit coordination methods are those where the agents share their observations but make decisions independently (Grocholsky, 2002). By sharing observations each agent is able to build the same common model about the state of the target, thus ensuring a higher amount of information over which it will take its independent decision (Cole, 2009). Therefore, the principal advantage of implicit coordination techniques is that they allow the agents to make decisions based on information that they have not themselves collected from the environment. Usually observations are shared and merged together using decentralised data fusion techniques (DDF); a well known and well studied set of techniques to efficiently estimate the state of a feature of interest through the means of Bayesian estimation (Bourgault et al., 2003, 2004; Hoffmann et al., 2006; Cole, 2009). The main idea is that each agent updates the estimated model of the target's position by considering also the observations made by other members of the team, which are shared by message-passing between the different agents (Grocholsky, 2002). Figure 2.6

Bordons, 2003; Garcia et al., 1989)

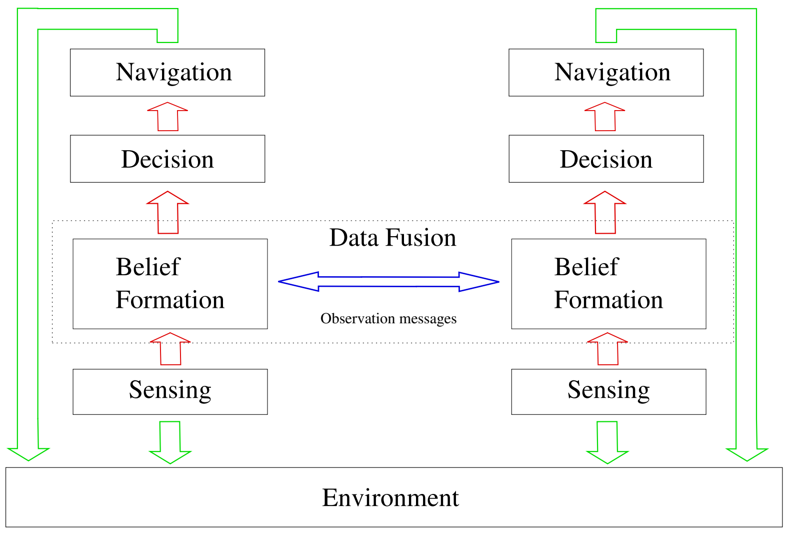


FIGURE 2.6: The architecture of an implicit coordinated approach (Cole, 2009)

illustrates an example of such type of approach. The architecture is essentially the same as the one in Figure 2.5, apart from the part that the agents share their observations in the belief formation shape.

Approaches to implicit coordination are generally the same as the non-coordinated one since the main difference between the two levels is the sharing of observations. Thus, they make decisions on which sub-region of the search area to explore next based on the same estimated model. Again, this model represents the uncertainty over the target location within the search space. Clearly such uncertainty is higher in areas that have not yet been explored, hence by keeping a common belief, the agents will aim for the unexplored sub-regions of the searching space (Cole, 2009). Examples of implicitly coordinated approaches include a global version of the greedy approaches presented in Section 2.3.1, where each agent, by gathering the observations of all the other agents, is able to build a more detailed model of the target's position considering also positions that it had not observed (Furukawa et al., 2006, 2007). In this context, receding horizon control is again applied together with greedy techniques, in order to have the agents observe the positions in the searching area with the highest level of uncertainty (Vidal et al., 2001; Hoffmann et al., 2006).

The main disadvantage of implicit coordination is that by deciding over the same knowledge the agents often end up taking the same actions and thus, explore the same areas. This redundant behaviour is clearly not advisable in a timely task, such as target search or target tracking (Cole, 2009). To perform effectively, the agents need to take into account the decisions of the other members of the team in order optimise the team performance. For this reason implicit coordination techniques are commonly considered to perform less well than their explicit counterpart. However, as we will see in more detail in the next section, explicit coordination often comes with a large computational overhead, thus implicit coordination has been shown to be useful in many robotics ap-

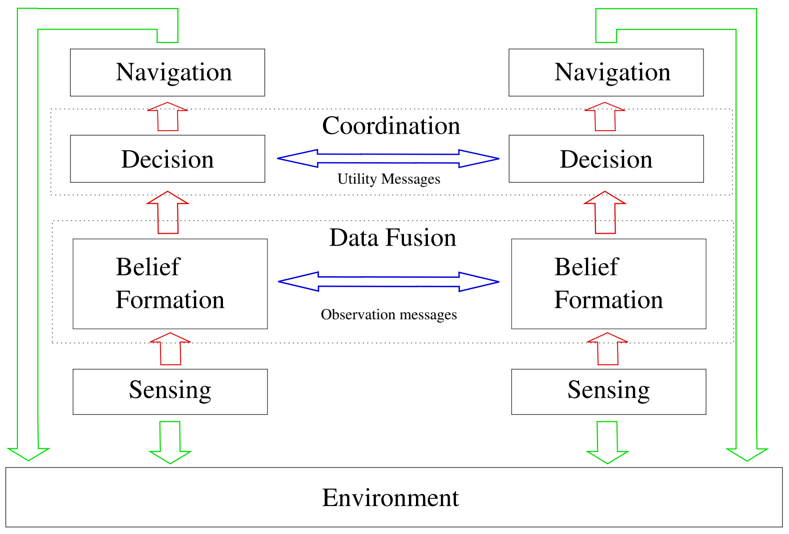


Figure 2.7: The architecture of an explicit coordinated approach (Cole, 2009)

plications, such as target search and target tracking (Hoffmann et al., 2006; Cole, 2009; Hollinger et al., 2009). For this reason, and similarly to the non-coordinated case, we will define an implicit coordination approach and use it as benchmark for our approach in Chapter 3.

2.3.3 Explicitly Coordinated Approaches

Explicit coordination methods are those where the agents share both their decisions and their observations. Figure 2.7 shows an example of such approach, where both the levels of information sharing are illustrated. The process of sharing observations is carried out in the same fashion as for the implicit coordination case.

In the explicit coordination case, the agents make their decisions by taking into account the actions of the other members of the team (Hollinger et al., 2009). However the main problem of explicit coordination techniques is that finding a joint optimal decision for a team of agents is exponential in the number of agents (Lesser et al., 2003), and thus the problem rapidly becomes infeasible as the number of members of the team grows larger (Farinelli et al., 2008; Hollinger et al., 2009).

In order to address the above-mentioned issues, the robotics community have explored techniques based on automated negotiation (Sujit and Ghose, 2005). In such approaches, the agents formulate a proposal by locally computing a joint decision over the next areas to observe given their current information about the target's state (i.e. their common belief) and about the proposals received by the other members of the team. After a fixed interval of time, the decision process is stopped and the proposal is sent to the other members of the team. The whole negotiation process is repeated until all the agents converge to the same joint decision. The main advantage is that the algorithm is any-time, thus it can provide an approximate solution whenever the algorithm is stopped, and the quality of such solution can be increased if the algorithm is given more time to converge (Dean and Boddy, 1988). Moreover the agents just need to negotiate with their closest team members, because they are the ones whose actions are most likely to be inter-dependent. Hence negotiation algorithms offer a robust solution against communication failure, and provide the possibility to scale up (Sujit and Ghose, 2005). However, the major drawback in this type of approach is that the time needed to converge to the optimal solution is in the worst case exponential in the number of agents, thus a long interval of time is required to reach a good solution.

This type of algorithm was further extended by Grocholsky (2002), where the problem is represented as a distributed optimisation problem, and solved using algorithm based on either the well known Jacobi or a Gauss-Seidel iteration. These algorithms iterate over a solution, and can provide any-time solutions. The main difference with the classic negotiation algorithm defined above, is the way in which the two algorithms update their current decisions. In more detail, Jacobi iteration allows every agent to sequentially make its decisions. Thus every agent makes a decision in a sequential order, waiting for the decisions of the agents that precede it. Conversely, Gauss-Seidel iteration allows every agent to make their decisions concurrently. Thus every agent makes a decision given the knowledge that it has and broadcasts its decision to the other agents. Both Gauss-Seidel and Jacobi iterations repeat until all the agents converge to the same solution.

Now, in both of these algorithms, an agent makes a decision considering the information about the current decisions of the other agents that it was able to collect. Moreover, each agent's knowledge about the other agents' decisions is not complete, and is going to be refined during the whole iteration process by sharing upgraded decisions. For these reasons, the agents are commonly said to play a best response to their neighbours' proposed decisions. That is, taking the best possible action (i.e. response) given the actions proposed by the other agents, and both Gauss-Seidel and Jacobi iterations are best response algorithms. However, for the same reasons as the ones mentioned for the negotiation algorithm, these approaches require a large amount of time in order to converge to a good solution (Grocholsky, 2002).

A similar approach was used in another study on the use of UAVs for search and tracking. In this case, the algorithm does not iterate, but locally calculates the joint decision, formulating the problem as a non linear optimisation problem, and then using pre-tuned fast solving algorithms. While it was acknowledged that this approach is sub-optimal, it was argued that this was necessary in a highly dynamic environment, since in the time required for the iteration to take place, the environment would have changed, rendering the iterated solution sub-optimal (Cole, 2009). However using a local optimisation approach requires each agent to consider all other agents in the team. Thus the algorithm is again exponential in the number of agents, and does not scale well as the number of such agents grows.

Many of the approaches to explicit coordination developed in the robotics community, share parallels with approaches developed in the multi-agent system (MAS) community. Here work has focused on studying ways to coordinate platforms such as mobile sensors deployed within an environment to solve tasks such as gathering information about some features of interest such as water contamination (Krause et al., 2008) or for target tracking (Lesser et al., 2003). Within these domains, the main requirements for coordination approaches are the same of the one described in Chapter 1 (i.e. adaptiveness, robustness, scalability and autonomy). Furthermore a mobile sensor presents features similar to UAVs, namely limited power, computational resources and communication range (Farinelli et al., 2008). Again, these approaches have focused mainly on a solving the problem in a decentralised fashion, hence avoiding central points of failure.

Within the MAS literature, these explicit coordination problems, are often represented as distributed constraint optimisation problems (DCOPs) and a number of algorithms exist to solve such DCOPs. In more detail, a class of optimal algorithms was proposed namely, OptAPO (Mailler and Lesser, 2004), ADOPT (Modi et al., 2005) and DPOP (Petcu and Faltings, 2005). OptAPO uses a partially centralised approach in which mediator agents compute solutions for portions of the problem, while ADOPT and DPOP formulate the constraints of the problem as a graph, and then exchange messages between nodes this graph. ADOPT is based decentralised version of the branch and bound algorithm, while DPOP is based on the generalised distributed law (GDL) class of algorithms (Aji and McEliece, 2000).

However, it has been argued that these algorithms are not suitable to address the coordination of mobile sensors because they do not address most of the above-mentioned requirements (Lesser et al., 2003). In particular, within OptAPO mediator agents may 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. Similarly, in ADOPT, the number of messages that agents exchange is exponential in the width of the tree, thus requiring again a large time interval to converge to a good solution. Finally, the size of the largest messages in DPOP can be exponential in the width of the tree, thus requiring again an unfeasible time to be computed (Farinelli et al., 2008).

On the other hand, a second class of approximate stochastic algorithms have also been proposed for solving DCOPs (Maheswaran et al., 2005; Fitzpatrick and Meertens, 2003). In these algorithms, each agent updates its state based only on the communicated (or observed) states of those local neighbours that influence its utility. For this reason, these approaches are well suited for large scale distributed applications, and in this context the “distributed stochastic algorithm” (DSA) (Maheswaran et al., 2005) is one of the most used as it has been shown to outperform the best response algorithm, defined at the beginning of this section (Farinelli et al., 2008; Chapman et al., 2011), and to perform well for searches for lost casualties in a simulated environment like the Robocup Rescue (Chapman et al., 2009). The main advantage of DSA over best response algorithms is that after calculating the best response action, DSA decides probabilistically to effectively take this action or not. Thus it might partially avoid the local minima problem described in Section 2.3.2. However, algorithms of this type often converge to poor quality solutions since agents do not explicitly communicate their utility for being in any particular state, but only communicate their preferred state (i.e. the one that will maximise their own utility) based on the current preferred state of their neighbours.

For the purposes of our research, we will use a state-of-the-art explicit coordination algorithm based on a best-response algorithm as a benchmark for the coordination algorithm that we will present in Chapter 3. We discuss next the principal contribution that we take from literature, namely the max-sum algorithm.

2.3.4 The Max-Sum Algorithm

As we have presented in Chapter 1, the max-sum algorithm belongs to the class of algorithms defined by the Generalised Distributive Law (GDL), a particular type of approximate message passing algorithms, well used and studied in the field of information theory (Aji and McEliece, 2000). Moreover, this algorithm is extremely relevant from our perspective, as it has been shown to meet most of the requirements presented, again, Chapter 1.

In this context, max-sum has been shown to generate solutions closer to the optimum than (for example) DSA or Best Response, while being robust against message loss and exhibiting a scalable computational and communication cost (Farinelli et al., 2008). Max-sum, as any GDL algorithm, exploits the factorisability of many optimisation problems (as presented in Section 2.4.2.4), to solve them in an effective and efficient manner. Particularly, the class of distributed constraint optimisation problems (DCOPs) introduced next, exhibit this characteristic (Farinelli et al., 2008).

2.3.4.1 DCOP Formalisation

The most general characterisation of a DCOP involves M agents, each controlling a single discrete variable x_j, $j \in [1, M]$. Constraints between agents are encoded as functions $U_i(\mathbf{x}_i)$ ($i \in [1, N]$) over these variables. The scope $\mathbf{x}_i \subseteq \mathbf{x}$ of constraint function U_i contains the variables of the agents over which the constraint is defined. The aim of the coordination problem is then to choose variable assignments that maximise the sum of the constraint functions:

$$U(\mathbf{x}) = \sum_{i=1}^{N} U_i(\mathbf{x}_i) \tag{2.1}$$

In order to use the max-sum algorithm the problem is encoded as a special bipartite graph called a factor graph, in which vertices represent variables and functions, and edges the dependencies between them.

2.3.4.2 Algorithm Characterisation

Max-sum defines two types of messages that are exchanged between variables and functions:

From variable x_j to function U_i:

$$q_{j\to i}(x_j) = \sum_{k\in M(j)\backslash i} r_{k\to j}(x_j) \tag{2.2}$$

where $M(j)$ represents the set of indices of the functions connected to variable x_j (i.e. the functions in which x_j occurs as an argument).

From function U_i to variable x_j:

$$r_{i\to j}(x_j) = \max_{\mathbf{x}_i\backslash x_j}\left(U_i(\mathbf{x}_i) + \sum_{k\in N(i)\backslash j} q_{k\to i}(x_k)\right) \tag{2.3}$$

where $N(i)$ represents the set of indices of the variables connected to function U_i.

Note that both $q_{j\to i}(x_j)$ and $r_{i\to j}(x_j)$ are functions of variable x_j. When the factor graph is acyclic, these messages represent the maximum aggregate utility possible over the respective components of the graph formed by removing the dependency between U_i and x_j, for each value $d \in D_{x_j}$ in the domain of variable x_j. Thus, in this case, the marginal function of each variable is calculated by:

$$z_j(x_j) = \sum_{i\in M(j)} r_{i\to j}(x_j) = \arg\max_{\mathbf{x}\backslash x_j}\sum_{i=1}^{N} U_i(\mathbf{x}_i) \tag{2.4}$$

after which the optimal assignment of x_j is found by:

$$a_j = \arg\max_{x_j} z_j(x_j)$$

An inherent shortcoming of the max-sum algorithm, however, is that it is not guaranteed to converge on cyclic constraint graphs, in which case it can perform arbitrarily poorly. This limits its applicability in our domain, which, as we have shown in Chapter 1, requires quality guarantees. The *bounded* max-sum algorithm, an extension to the standard max-sum algorithm, addresses this shortcoming, by pruning the constraint graph to a tree. In so doing, it is capable of providing performance guarantees on the computed solutions (Rogers et al., 2011).

2.3.4.3 The Bounded Max-Sum Algorithm

When the factor graph is cyclic, the straightforward application of max-sum is not guaranteed to converge. However, by pruning edges such that an acyclic sub-graph of the factor graph is obtained, a bounded approximation can be derived (Rogers et al., 2011). More specifically, here, the goal is to compute a variable assignment $\tilde{\mathbf{a}}$ in the acyclic factor graph, such that:

$$U^* = \sum_{i=1}^{N} U_i(\mathbf{a}_i^*) \leq \rho \sum_{i=1}^{N} U_i(\tilde{\mathbf{a}}_i)$$

where $\mathbf{a}^*$ is the optimal solution of the cyclic factor graph, U^* 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 x_j and U_i is defined as its weight w_{ij}, which is computed by:

$$w_{ij} = \max_{\mathbf{x}_i \setminus x_j} \left[\max_{x_j} U_i(\mathbf{x}_i) - \min_{x_j} U_i(\mathbf{x}_i) \right] \tag{2.5}$$

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

$$\tilde{\mathbf{a}} = \arg\max_{\mathrm{x}} \sum_i \min_{\mathbf{x}_i^c} U_i(\tilde{\mathbf{a}}_i)$$

where $\mathbf{x}_i^c$ is the set of variables that were eliminated from the scope of function U_i, corresponding to the edges that were pruned from the factor graph.

The approximation ratio ρ is now given by:

$$\rho = 1 + (\tilde{U}_m + W - \tilde{U})/\tilde{U} \tag{2.6}$$

where W is the sum of the weights of the pruned edges, $\tilde{U}_m = \sum_i \min_{\mathrm{x}_i^c} U_i(\tilde{\mathbf{a}}_\mathbf{i})$ is the value of the solution of the pruned problem and $\tilde{U} = \sum_i U_i(\tilde{\mathbf{a}}_\mathbf{i})$ is the value of the solution of the overall approximated problem. Thus, an upper bound on the optimal solution can be computed as follows:

$$\tilde{U}_m + W \geq U^*$$

The application of the max-sum algorithm to the coordination of mobile sensors to monitor spatial phenomena (Stranders et al., 2009) was particular interesting, because it demonstrated that the max-sum algorithm is applicable to more complex problems than those described in Farinelli et al. (2008). Additionally, a number of similarities between the search problem and the spatial phenomena monitoring problem, such as gathering information or cooperatively explore an environment, are apparent.

These similarities motivate the use of the max-sum algorithm as a starting point in our work. Specifically, in the rest of this work, we will first extend max-sum in order to tackle the problem of on-line, decentralised coordination of UAVs on a target search domain, which cannot be performed effectively with existing techniques. Second, we will build upon max-sum and its corresponding bounded extension to the multi-objective coordination approach that we are going to present in Chapter 4.

2.4 Multi-Objective Optimisation

As noted in Chapter 1, current decision making approaches for teams of UAVs fall short in terms of handling problems involving multiple objectives. Indeed, this line of research lacks principled approaches that can address problems such as the search and track problem discussed in Section 2.2.2, while simultaneously optimising other objectives, such as managing energy resources as mentioned in the scenario presented in Chapter 1. However, handling multiple objectives is a fundamental requirement in order to enhance the deployment of UAVs for real world operations, and therefore constitutes a key challenge of our research.

Thus, in this section, we review literature on multi-objective optimisation. As mentioned in the previous section, coordination for information gathering problems consists of optimising a global utility function, shared among different agents, in a decentralised way. In a multi-objective setting, there is a set of such utility functions, therefore, we discuss here existing approaches able to address this issue. We will, initially, formalise two key concepts in order to characterise solutions to multi-objective optimisation problems. Next we introduce the current state-of-the-art approaches and then, we evaluate them in our framework.

2.4.1 Problem Formalisation

We introduce in this section some formal concepts that we are going to use throughout the thesis. Thus, we define a multi-objective optimisation problem (MOOP) as the problem of simultaneously optimising a set of incommensurate *objective functions*. Considering the scenario introduced in Chapter 1, a multi-objective problem involving three objectives could be: searching for lost targets, tracking those that have already been found, and managing the energy resources. Nevertheless, in order to keep our exposition as clear as possible, we will consider the problem of maximising k objective functions, defined over a set $\mathbf{x} = \{x_1, \ldots, x_M\}$ of M discrete variables, where each x_j takes values in a discrete domain $D_{x_j} = \{d_j^1, \ldots, d_j^{|D_{x_j}|}\}$. A solution to a MOOP is, then, an assignment $\mathbf{a}^* = \{(x_1 = d_1^1), \ldots, (x_M = d_M^M)\}$ of values to variables, such that:

$$\mathbf{a}^* = \arg\max_{\mathbf{a} \in D_\mathbf{x}} \mathbf{U}(\mathbf{x}) = [U^1(\mathbf{x}), \ldots, U^k(\mathbf{x})]^T \tag{2.7}$$

where $D_\mathbf{x} = \times_{j=1}^{M} D_{x_j}$ is the domain of variables $\mathbf{x}$. Here, each objective function U^i can be defined over a subset $\mathbf{x}_i \subseteq \mathbf{x}$ 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 the functions are incommensurable, it is possible that multiple assignments satisfy Equation 2.7. For example, consider three assignments $\mathbf{a}_1$, $\mathbf{a}_2$ and $\mathbf{a}_3$, such that $\mathbf{U}(\mathbf{a}_1) = [4, 5]$, $\mathbf{U}(\mathbf{a}_2) = [4, 3]$, and $\mathbf{U}(\mathbf{a}_3) = [6, 3]$. Clearly we have that $[4, 5]$ and $[6, 3]$ are larger than $[4, 3]$. Thus, $\mathbf{a}_2$ does not satisfy Equation 2.7. Moreover, $[4, 5]$ and $[6, 3]$ are not comparable. Indeed, Equation 2.7 involves the optimisation of sets of *partially-ordered* assignments. Thus, to characterise the optimal solutions of a multi-objective optimisation problem, we use the concept of *Pareto optimality*, the most used solution concept for multi-objective problems:

Definition 2.1. Pareto Optimality (Pareto, 1906): An assignment $\mathbf{a}^* \in D_\mathbf{x}$ is *Pareto optimal* iff there does not exist another assignment $\mathbf{a} \in D_\mathbf{x}$, such that $\mathbf{U}(\mathbf{a}) \geq \mathbf{U}(\mathbf{a}^*)$, and $U^i(\mathbf{a}) > U^i(\mathbf{a}^*)$ for at least one objective function.

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

Definition 2.2. Non-dominance (Pareto, 1906): A vector $\mathbf{U}(\mathbf{a}^*) \in D_\mathbf{x}$ is *non-dominated* iff there does not exist an assignment $\mathbf{a} \in D_\mathbf{x}$, such that $\mathbf{U}(\mathbf{a}) \geq \mathbf{U}(\mathbf{a}^*)$, with at least one $U^i(\mathbf{a}) > U^i(\mathbf{a}^*)$. Otherwise, $\mathbf{U}(\mathbf{a}^*)$ 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. In the remainder of this document, we will refer to the set of Pareto optimal assignments as **PO**.

2.4.2 Multi-Objective Approaches

Multi-objective optimisation has been studied principally in the fields of operational research and artificial intelligence. Two principal categories of approaches have been proposed, namely approximate and optimal approaches, and they have both been used in many areas such as combinatorial auctions, flight formation problems and mobile robotics (Rollón, 2008; Inalhan et al., 2002; Mouaddib et al., 2007).

Multi-objective approaches can be categorised into four different classes of techniques, each of which constitute the state-of-the-art for multi-objective optimisation. We review here these four classes, also considering the type of solutions that they provide. More specifically we will discuss two classes of approximate approaches, namely evolutionary algorithms and scalarisation techniques, coming from operational research and two classes optimal of approaches, heuristic search algorithms and problem decomposition methods.

2.4.2.1 Evolutionary Approaches

The principal type of approximation methods developed to solve multi-objective optimisation problems are evolutionary algorithms (Coello Coello, 2006; Khare et al., 2003). Such approaches, also referred as population-based meta-heuristics (Zitzler et al., 2000), maintain a set of solutions (the population) and try to evolve the population towards the Pareto optimal set of solutions, based on a fitness function that evaluates the quality of each element of the population. Various techniques have been proposed in literature in order to extend evolutionary algorithms to the multi-objective domain, while guaranteeing empirically to retrieve solutions relatively close to the Pareto optimal set (Zitzler et al., 2000). Well known examples of these algorithms include the vector evaluated genetic algorithm (VEGA), the multi-objective genetic algorithm (MOGA) and the non-dominated sorting genetic algorithm (NSGA) (see Marler and Arora (2004) for a detailed review).

However, such approaches do not provide guarantees over the quality of the solutions they recover. Moreover, while some trivial parallelisation approaches for evolutionary algorithms have been proposed in literature, which consists in optimising the different objectives of the problem in a parallel fashion (Talbi et al., 2008), multi-objective evolutionary algorithms are essentially centralised (Marler and Arora, 2004), and therefore they would fail whenever a component failure occurs. Thus, such approaches are not suitable for our domain.

2.4.2.2 State Space Search Approaches

State space search has traditionally been one of the fundamental optimal problem solving tools in artificial intelligence (Russel and Norvig, 1995). The state space, represents the different configurations that the problem can assume, and it forms a weighted graph where nodes represent states and edges represent the transition from a state to another triggered by performing an action in a specific state. Each taken action has an associated cost. The task is then to compute the best path from the initial state to a goal state given the weights of the graph (Russel and Norvig, 1995)).

Within this domain, two well known best-first optimal algorithms, A* (Hart et al., 1968, 1972) and iterative deepening A* (Korf, 1985), have been extended to the multi-objective case. The main idea is to transform the weights of the edges of the state space graph into vector of weights, one weight for each objective, (Stewart and White, 1991; Harikumar and Kumar, 1996). Clearly, such vectors are partially ordered, therefore the algorithm works by computing the set of non-dominated best paths, according to a given path function.

Unfortunately, such approaches fall short in meeting many of the requirements mentioned in the Chapter 1. Specifically, the states space in the multi-objective context increases exponentially, therefore, exploring it to retrieve the entire set of Pareto optimal solutions requires a large amount of time (Stewart and White, 1991). This increase in the number of states also prevents these approaches from being able to handle more than 2 or 3 different objectives (Dasgupta et al., 1999). Finally, all these approaches are centralised and, thus far, no decentralised, nor parallel techniques have been developed. Moreover, such approaches do not meet the adaptiveness requirement that we introduced in the previous chapter. Indeed, in dynamic problems such as information gathering tasks, it is not possible to compute a complete state space, because most of the information required to build it, is not available beforehand. For these reasons, such approaches are not suitable for our target-searching domain and are not going to be considered further in this work.

2.4.2.3 Scalarisation Approaches

Scalarisation approaches are well known approximation techniques that convert the multi-objective problem into a standard single objective problem which is then solved using standard single objective solution techniques (Marler and Arora, 2004; Johannes, 2004).

In more detail, these methods define a scalarisation function that combines the different objectives of a problem into a single objective function, and then iteratively re-apply a standard single objective optimisation technique, such as linear or integer programming (Marler and Arora, 2004), to the original problem. In each of these iterations, the parameters characterising the scalarisation function (i.e. the weights of the different objectives) are changed. The reason for this is that a single objective problem is characterised by one optimal solution, while a multi-objective problem can have many. Thus, in order to retrieve the entire Pareto optimal set, which is a common requirement for multi-objective optimisation problems, the iteration procedure is adopted. Examples of scalarisation methods include the aggregation of the different objectives via a weighted sum, the optimisation of one objective function while bounding all the others and the definition of a ranking between the different objective functions based on a predetermined function that measures the importance of each of these objectives (Ehrgott and Gandibleux, 2000).

However, the entire set of Pareto optimal solutions can be retrieved only under some strong assumptions characterising the multiple objectives of the problem (i.e. they represent a convex set). If this not the case, nothing can be said about the set of solutions retrieved. Indeed, just some or even none of them, can belong to the Pareto optimal set.

Consequently, these approaches are not suitable to be used in the target search domain. First, our domain is not guaranteed to be convex, therefore by using these approaches we would not be able to provide guarantees on the quality of our solutions. Second, scalarisation approaches require a significant amount of time in order to retrieve all such solutions (Marler and Arora, 2004), therefore, they do not meet the timeliness requirement described in Chapter 1.

2.4.2.4 Approaches based on Problem Decomposition

A fourth type of approach, pivotal to the algorithm that we present in Chapter 4, focuses on solving multi-objective optimisation problems in a more effective fashion by exploiting their inherent structure (Rollón, 2008). This type of approach has been widely exploited for solving single-objective optimisation problems (Aji and McEliece, 2000; Farinelli et al., 2008). The essential idea behind this class of algorithms is to exploit a common characteristic of many optimisation problems in order to decompose them in smaller subproblems and solve them in a quicker and more efficient fashion (Aji and McEliece, 2000). Well known examples of these problems are constraint optimisation problems, constraint satisfaction problems and distributed constraint optimisiation problems (Dechter, 2003; Bistarelli et al., 1997; Modi et al., 2005).

In more detail, the structure of these problems requires a specific algebraic characteristic in order to be decomposable. Specifically, given the objective function $U : \mathbf{D} \rightarrow R$, the principal requirement for an optimisation problem to be factorisable, is that the codomain R of U is a *commutative semi-ring*. Such a commutative semi-ring is defined as:

Definition 2.3. A **commutative semiring** is a set R, together with two binary operations called "$\oplus$" and "$\otimes$", which satisfy the following three axioms:

1. The operation "$\oplus$" is associative and commutative, and there is an additive identity element called "0" such that $k \oplus 0 = k$ for all $k \in R$ (This axiom makes $(R, \oplus)$ a commutative monoid).

2. The operation "$\otimes$" is also associative and commutative, and there is a multiplicative identity element called "1" such that $k \otimes 1 = k$ for all $k \in R$ (Thus $(R, \otimes)$ is also a commutative monoid).

3. The distributive law holds, i.e.,

$$(a \otimes b) \oplus (a \otimes c) = a \otimes (b \oplus c) \tag{2.8}$$

for all triples (a, b, c) from R

The main reason why this property is fundamental is the third axiom of the previous definition (Aji and McEliece, 2000). Specifically, an algorithm optimising a function whose co-domain is a commutative semiring can exploit the distributive law to reduce the number of calculations necessary to solve the problem (it is a generalisation of the simple fact that calculating $a \otimes (b \oplus c)$ requires only 2 operations, while $(a \otimes b) \oplus (a \otimes c)$ requires 3 operations).

Recent work on multi-objective optimisation has focused on ways to characterise such problems as semiring optimisation problems. In this vein, Rollón (2008) introduces one of the contributions pivotal to our work, the semiring multi-objective optimisation problem. Specifically, Rollón (2008) describes a framework to aggregate k different single-objective semiring optimisation problems into a proper semiring multi-objective optimisation framework. This approach was employed in order to extend a well known algorithm exploiting problem decomposition, the bucket elimination algorithm (BE) (Dechter, 2003), for solving multi-objective problems (Rollón and Larrosa, 2006).

More specifically, the bucket elimination algorithm works by iteratively eliminating the variables of the problem and deducing the effect of these eliminated variables from the global objective function on the problem. The elimination of the last variable produces a constant that constitutes the optimal solution of the problem. Now, to extend the problem to multiple objectives, Rollón and Larrosa (2006) extend the two fundamental operators of BE, namely the sum between functions and the variable elimination operator (also referred as the variable marginalisation operator (Aji and McEliece, 2000)) to the multi-objective domain, and shows how the resulting algorithm, multi-objective bucket elimination (MO-BE), retrieves the entire Pareto optimal set of solutions for arbitrary multi-objective optimisation problems. However, MO-BE is, again, not suitable for our problem, because it is a centralised algorithm, whose empirical evaluation has shown that it does not scale to more than two objectives (Rollón and Larrosa, 2006).

However, as pointed out by Rollón (2008) bucket elimination algorithms and GDL algorithms are both problem decomposition methods, and thus share many similarities, such as the way the function are aggregated or the way the variables are marginalised. For this reason, we will use the two main operators of MO-BE (i.e. the sum and the marginalisation operators) in order to build our algorithm in Chapter 4

2.5 Summary

In this chapter we reviewed the scientific literature relevant to our work. We started by introducing the current state-of-the-art in unmanned aerial vehicles (UAVs), the particular robotic platforms that we will consider in our work. We then introduced literature concerning the target search problem. In particular, we started by analysing literature on canonical target search, discussing how the problem is simplified in much of the literature by relaxing most of its features, including discretising the search area (Hoffmann et al., 2006), simplifying the motion of the agents (Hespanha et al., 1999; Vidal et al., 2001) and simplifying the sensing capabilities of the agents (Antoniades et al., 2003). We then presented the search and track problem, which builds upon search, but adds the additional dimension of tracking targets that have already been found (Furukawa et al., 2006).

Next, we examined literature on existing coordination approaches. Initially, we defined the three common levels used to differentiate coordination approaches (Cole, 2009). In more detail, we depicted the non-coordinated level as a lower bound to measure more sophisticated coordination approaches commonly used in the literature (Bourgault et al., 2003; Hollinger et al., 2009; Cole, 2009). We further introduced the implicit coordination level as a valid trade-off between the quality and the feasibility of the solutions to the coordination problem, where agents share their observations thus increasing their knowledge about the problem to solve (Hoffmann et al., 2006; Hollinger et al., 2009). Finally, we introduced the literature relevant to our work, namely explicit coordination approaches.

We then introduced the max-sum algorithm, a particular type of approximate message passing algorithm, belonging to the Generalised Distributed Law (GDL) class of algorithms (Aji and McEliece, 2000). Max-sum works on a constraint graph representing the agents (i.e. the vertices of the graph) and the different relationships between them (i.e. the edges of the graph). An inherent short-coming of the max-sum algorithm, however, is that it is not guaranteed to converge on cyclic constraint graphs, in which case it can perform arbitrarily poorly (Farinelli et al., 2008). This limits its applicability in safety-critical domains. The bounded max-sum algorithm addresses this shortcoming, by pruning the constraint graph of an arbitrary problem to a tree, and then by running the max-sum algorithm over the cycle-free constraint graph (Rogers et al., 2011). In so doing, it is capable of providing performance guarantees on the computed solutions (Rogers et al., 2011). For all the above reasons, we choose the max-sum algorithm as the foundation over which we are going to build our coordination algorithms.

Now, the applicability of the max-sum algorithm to a complex problem such as coordinating agents for target search, has not yet been studied. Moreover, many issues arise when dealing with a more realistic setting, such as communication limitations and uncertainty about the targets and the state of the team members. For these reasons, we present in the next chapter our first contribution, a novel decentralised coordination approach for coordinating teams of UAVs for target-search. In particular, we apply, for the first time, the max-sum algorithm to a complex and realistic problem, in which both the area of interest, the UAVs and the target are represented with complex motion and sensors models (Bourgault et al., 2003, 2004; Cole, 2009). Specifically, we will use the formalisation of the problem depicted in Bourgault et al. (2003) and we define each UAV as having a fixed camera (Xu and Zhang, 1996), that allows it to observe the ground. The information provided by each observation is fused with the one provided by the others UAVs, by using a decentralised data fusion process as per the one used in Bourgault et al. (2003) and Cole (2009).

Subsequently, we discussed literature related to multi-objective optimisation. We first introduced the fundamental concepts, the notion of Pareto optimality and the dominance that characterise the solutions of multi objective problems (Pareto, 1906). We then reviewed the state-of-the-art-approaches for solving these specific problems. In more detail, we showed that approximate approaches such as evolutionary algorithms (Zitzler et al., 2000) and scalarisation techniques (Marler and Arora, 2004), respectively, require too much time to converge to a solution or are not able to provide quality guarantees over such solutions and therefore do not satisfy the requirements defined in Chapter 1. Similarly, optimal approaches such as state space search (Stewart and White, 1991) and approaches based on problem decomposition, such as the bucket elimination algorithm (Rollón, 2008) are not adaptive to the dynamism of an information gathering task, and they are also centralised. Thus, they are, again, not suitable for our problem. However, since the max-sum shares some similarities with the bucket elimination algorithm, we will use its two main operator—the maximum and the variable elimination operator— in order to build the multi-objective decentralised coordination algorithm that we are going to present in Chapter 4. Moreover, since we require an optimal algorithm to measure the quality of the solutions retrieved by our multi-objective algorithm, we will employ a brute-force multi-objective approach, that searches the entire space of solutions of a problem, in order to retrieve its entire set of Pareto optimal solutions.

Then, in Chapter 4 we present our second contribution; a novel bounded decentralised coordination algorithm for computing approximate solutions over multi-objective optimisation problems. This algorithm extends the bounded max-sum algorithm for coordination and proceeds in three phases. In the first phase, we extend the approach defined in Rogers et al. (2011) to the multi-objective case, and prune the constraint graph of the problem into a tree. In the second phase, we extend max-sum to the multi-objective domain by building over the multi-objective bucket elimination algorithm (Rollón, 2008) and over the standard max-sum algorithm (Farinelli et al., 2008). Finally, in the third and final phase, our algorithm selects a solution, between the one retrieved during the previous phase, based on a logical criterion.

Chapter 3

Explicit Coordination using the Max-Sum Algorithm

We present in this chapter our first contribution. We focus here on developing a coordination algorithm for the canonical target-search problem. By doing so, our aims are twofold. First we develop a novel decentralised coordination technique for teams of UAVs, based on the max-sum algorithm. Second, by applying the max-sum algorithm to more realistic settings than those in which it was applied before, we are able to verify which of the requirements that we presented in Chapter 1 it satisfies and in what degree. In particular, the approach that we present in this chapter aims to address the requirements of timeliness, adaptiveness, robustness, autonomy and scalability.

Thus, in the remainder of this chapter, we present in Section 3.1 the model of the target search problem that we consider, introducing the motion models of all the agents, and the sensor model of the camera. In Section 3.2, we define the coordination mechanism that we developed in order to solve the problem. In particular, in Section 3.2.1 we set up the decentralised data fusion approach used to maintain a common belief for every agent and in Section 3.2.2 we present our decentralised coordination approach based on an adaptation of the max-sum algorithm. Finally in Section 3.3 we define the experiments we run, by introducing the benchmarks, the methodology, and the results obtained. Section 3.4 summarises the chapter.

3.1 Problem Model

In this section, we introduce the model of the search task. Fundamentally, the problem is to coordinate the motion, and hence observations, made by a team of UAVs so as to search for a target in a timely manner. We define a continuous search area A where the UAVs operate to search for the target. Each UAV maintains an internal representation of the area A by discretising the area into a grid G with a resolution depending on the setting of the problem. Each cell of the grid G represents a specific rectangular area within the search area. We model the continuous time system using small, discrete time steps, and assume that all the UAVs have time synchronisation (e.g. through GPS time). We define the search task by modelling the motion of the agents, the target and the UAVs, and the sensor model.

3.1.1 Agent Model

The scenario mentioned at the beginning of this section suggests two types of agents, the UAVs and the target.

3.1.1.1 Target Motion Model

We assumes the target moves following a simple probabilistic Markov motion model, a standard, and widely used, approach for representing the uncertainty over the target's motion (see Section 2.2). The state of the target at time k is defined as $x_k^T = (i, j)$, where (i, j) are the coordinates of the grid cell that contains the target. The probability of the target transitioning to another cell is modelled as:

$$P\left(x_{k+1}^T | x_k^T\right) = \frac{1}{\left|Adj\left(x_k^T\right)\right|} \tag{3.1}$$

where $Adj\left(x_k^T\right)$ is the set of cells adjacent to x_k^T as well as x_k^T itself.

3.1.1.2 UAV Motion Model

The team of UAVs is formally defined as a set S of agents. Again, we adopt a standard and well known, model in order to represent the vehicles' motion (see Section 2.2):

$$\dot{x} = V \cos\psi \tag{3.2}$$
$$\dot{y} = V \sin\psi \tag{3.3}$$
$$\dot{z} = 0 \tag{3.4}$$
$$\dot{\psi} = \frac{g \tan\phi}{V} \tag{3.5}$$

where V is the UAV velocity, g is the acceleration due to gravity, ψ is the UAV heading and ϕ is the UAV bank angle, which is limited to some maximum value $|\phi| \leq \phi_{max}$. As defined in Cole (2009), we assume that the velocity of the UAV remains constant at the cruise speed, which was taken as $25m/s$, and that the maximum bank angle is 25 degrees.

3.1.2 Sensor Model

Each UAV uses a fixed, downward pointing camera to detect the target. The camera is assumed to capture one frame per second. The primary interest in the sensor model is to characterise the footprint of the camera, as well as the probability of detecting a target within that footprint. This particular sensor model has been chosen because it is the most common type of sensor used within the target search domain (as discussed in Section 2.2).

Formally, we denote sensor observations by the m^{th} UAV at the k^{th} time step as z_k^m, where z_k^m can take on one of two values, D_k^m, representing a target detection event, or $\bar{D}_k^m$, representing a no-detection event. We further define z_k as the net observations by all UAVs.

Moreover, we define a matrix o_k^m, where the $(i,j)^{th}$ element, denoted by $o_k^m(i,j)$, represents the probability of the sensor on UAV m not detecting the target, conditional on the target being at the $(i,j)^{th}$ cell:

$$o_k^m(i,j) = P\left(z_k^m = \bar{D}_k^m | x_k^T\right) \tag{3.6}$$

Naturally, $P\left(z_k^m = \bar{D}_k^m | x_k^T\right) = 1 - P\left(z_k^m = D_k^m | x_k^T\right)$.

In order to model $P\left(z_k^m = D_k^m | x_k^T\right)$, we first characterised the footprint of the camera, which was modelled as a fixed camera (Xu and Zhang, 1996). As a consequence of this, the footprint can be easily computed by making a flat Earth assumption, commonly adopted in most of the related literature (Hollinger et al., 2009), and by simple geometric arguments. Figure 3.1 shows an example of a camera footprint[1].

When the quadrilateral defined by the points $\overrightarrow{P_i}$ for $i \in [1,4]$ in Figure 3.1 is overlaid onto the grid G, the probability of detecting the target, that is $P\left(z_k^m = D_k^m | x_k^T\right)$, is assumed to be linearly proportional to the ratio of the area of the cell covered by the quadrilateral to the total area of the cell, multiplied by a term α that models the range-dependent characteristics of the sensor. In this case, the range dependent characteristics were modelled as:

$$\alpha = \exp\left(-\frac{R}{R_0}\right) \tag{3.7}$$

where R is the range from the sensor to the cell in question and R_0 is a constant term that was tuned to model the range-dependency of the sensor. Therefore, the value of $P\left(z_k^m = D_k^m | x_k^T\right)$ for each cell in the grid G has a value varying from 0, when the cell is not within the footprint and α when the cell is completely covered by the footprint. It can be noted that this model accounts only for false negatives, where the sensor fails

[1]The authors would like to acknowledge V. Scordamaglia' s MATLAB function *Trajectory and Attitude Plot Version 2* as the source of the aircraft model used in a number of illustrations in this document. It can be obtained from `http://www.mathworks.com/matlabcentral/fileexchange/4572-trajectory-and-attitude-plot-version-2`

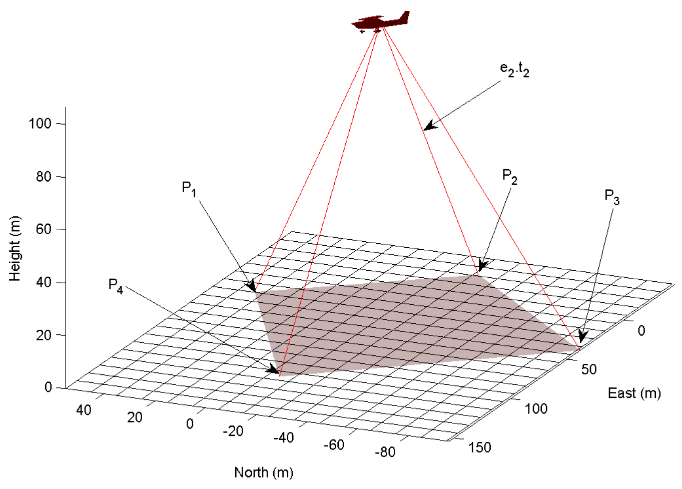


FIGURE 3.1: Illustration of camera footprint.

to detect a target that is present in the sensor field of view, but not for false positives, where the sensor reports a detection when the target is not present in the field of view.

The UAVs share these observations with each other to maintain a consistent belief of the distribution over the state of the target across the UAVs. Assuming that the observations by the UAV sensors are conditionally independent, then

$$P\left(z_k = \bar{D}_k | x_k^T\right) = \prod_{m=1}^{|S|} P\left(z_k^m = \bar{D}_k^m | x_k^T\right) \tag{3.8}$$

where $\bar{D}_k = \bar{D}_k^1 \cap ... \cap \bar{D}_k^{|S|}$. Defining $o_k(i,j) = P\left(z_k = \bar{D}_k | x_k^T\right)$, the above equation means that:

$$o_k = \prod_{m=1}^{|S|} o_k^m \tag{3.9}$$

3.1.3 Communication Model

In order reproduce in a more effective way the conditions of real world scenario, we will adopt two types of models to allow communication between UAVs:

- unlimited communications, in terms of distance, band-width and an error-free channel.
- range limited communications, that is communication between two UAVs is only possible when the Euclidean distance between them is below some threshold.

In particular unlimited communication will be, initially, used in order to compare specifically the performance of the different coordination approaches, as will be shown later on this section. Subsequently such an assumption will be disregarded, and the range limited communication model is going to be used. This model was chosen because of empirical evidence showing that it is valid for open environments (Mong-Ying et al., 2006).

It should be noted that the limiting of communications will make the maintenance of a common PDF amongst the team of UAVs (i.e. data fusion) a much more challenging task, for which the simple sharing of observations is inadequate. However it is not the purpose of this thesis to consider data fusion techniques to address this problem, thus in our specific case, with limited communication, the agents will no longer be able to maintain a common belief.

3.2 The Coordination Approach

In this section, we outline the framework for coordinating the sensor platforms such that they can collectively search for a target. We first introduce the data fusion methodology (see Section 2.3.2), and then the coordination approach we used. Figure 3.2 illustrates the same type of explicit coordination approach as illustrated by Figure 2.7, however, the structure, here, characterises our approach in the case of a single agent. In more detail, Figure 3.2 shows the different messages that are sent and received by the agent and how such messages are used in order to make a decision.

3.2.1 Bayesian Estimation

In this work, the probabilistic belief of the target's position over the grid G is defined as a matrix P_k^T, with each element of the matrix representing the probability of the target being in the corresponding cell in G at time k:

$$P_k^T\left(i,j\right) = P\left(x_k^T = (i,j)\right) \tag{3.10}$$

The estimation process involves two steps: the update step that fuses observations into the belief, and the prediction step that propagates the belief to account for the dynamic nature of the target. These steps constitute the belief information module depicted in Figure 3.2, and are described in detail in the following sub-sections.

A series of snapshots showing the changes in the distribution over the state of the target as the estimation process is carried out is shown in Figure 3.3. In more detail, the figure shows six different snapshots of an exploration task being done by a team of three UAVs (i.e. one blue, one green and one black). The six different snapshots, taken at different time-steps, show how the distribution over the state of the target (i.e. the red shape in the Figure) is changed due to the actions of each UAV. More precisely, Figures 3.3(a)

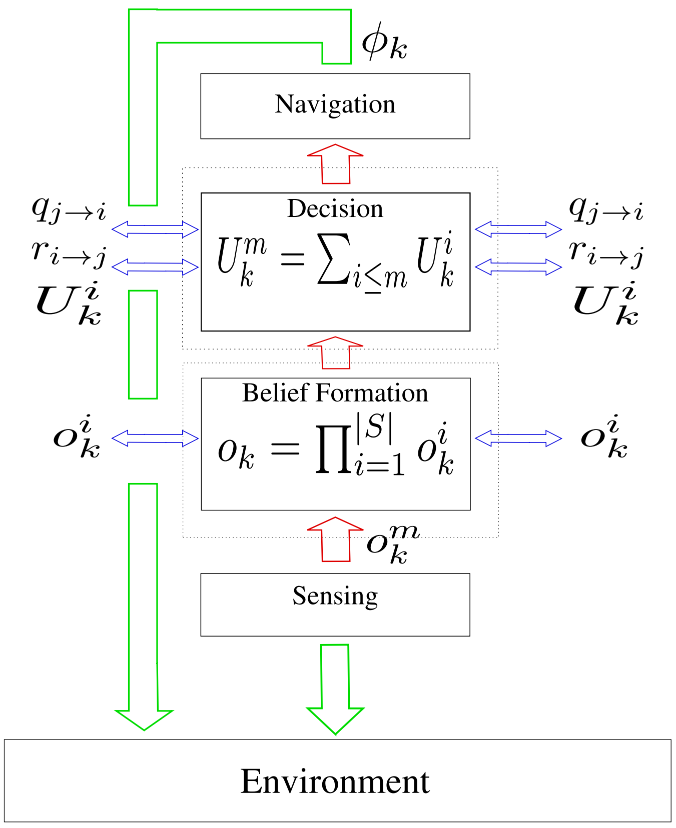


FIGURE 3.2: An illustration of the coordination approach.

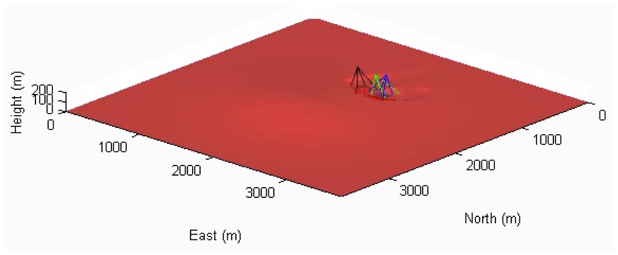
and 3.3(b) show the beginning of the simulation, in which the UAVs start the task at the same simulation, and begin searching while trying to avoid exploring the same areas (which testifies the coordinated behaviour). Figures 3.3(c), 3.3(d) and 3.3(e) depict the UAVs while searching the centre of the area, which is characterised by a high probability of detecting the target. Finally, Figure 3.3(e) shows the end of the simulation, where the pre-defined threshold has been reached.

3.2.1.1 Update

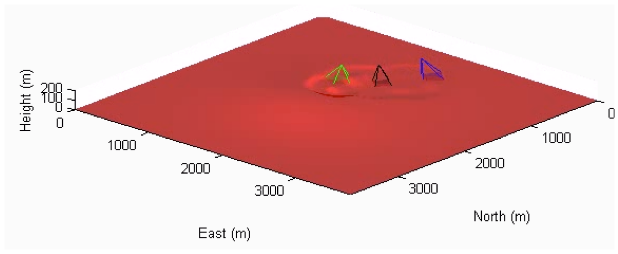
We adopted the same Bayesian update equation as used in previous work (Bourgault et al., 2004; Furukawa et al., 2006) to fuse the observations made by the UAVs into their belief of the state of the target. Thus,

$$P\left(x_{k+1}^T | z_{k+1}^m, \dots, z_1^m\right) = \frac{1}{C_1} P\left(x_{k+1}^T | z_k^m, \dots, z_1^m\right) P\left(z_{k+1}^m | x_{k+1}^T\right) \tag{3.11}$$

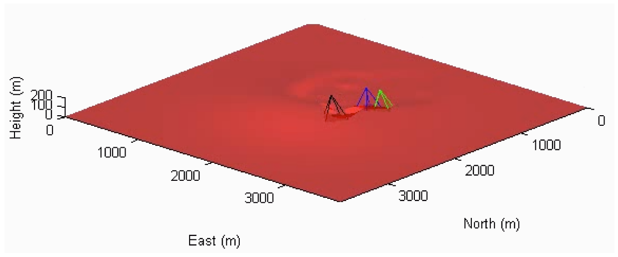
where C_1 is a normalising constant to ensure that the probability distribution function (PDF) integrates to unity, and is equal to $P\left(z_{k+1}^i | z_k^m, \dots, z_1^m\right)$. Here, $P\left(z_{k+1}^i | x_{k+1}^T\right)$ is the sensor model, and takes on the value of $P\left(z_k^m = D_k | x_k^T\right) = 1 - o_k^m$ when the target is detected, and $P\left(z_k^m = \bar{D}_k^m | x_k^T\right) = o_k^m$ in the case where the target is not detected.



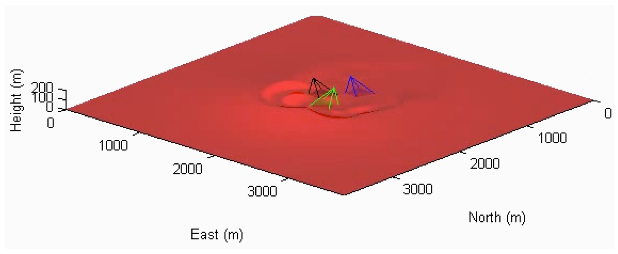
(a) $t = 0$: The UAVs start at the same position



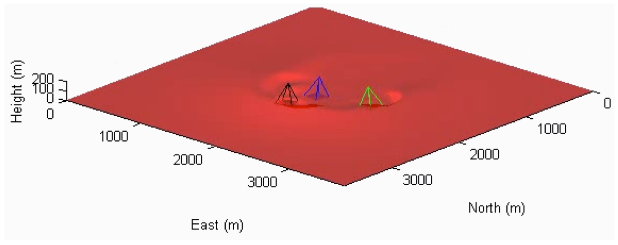
(b) t=20: Coordinated search



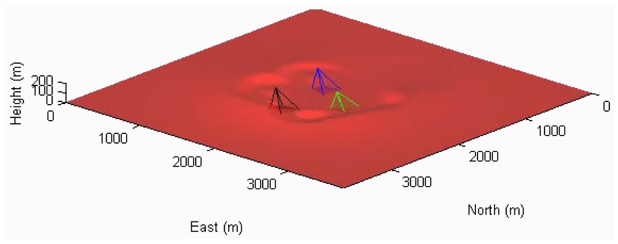
(c) $t = 40$: Non-coordinated search



(d) $t = 60$: The UAVs reach the area with a high probability

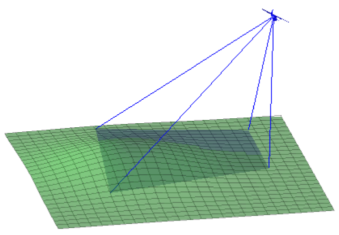


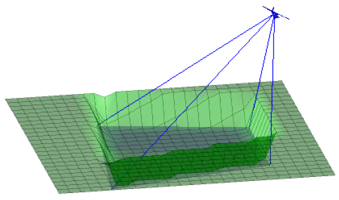
(e) $t = 80$: Coordinated search

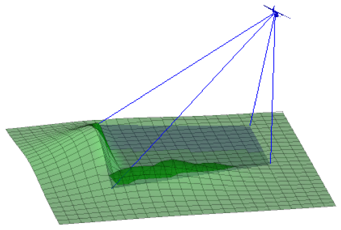


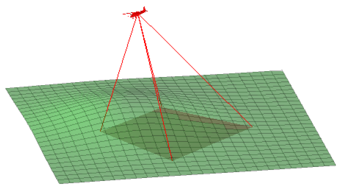
(f) $t = 100$: The threshold has been reached

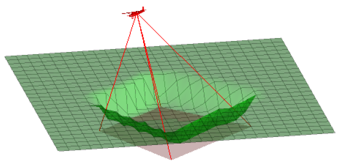
FIGURE 3.3: A series of snapshots showing the changes in the probability distribution over the state of the target as a team of three UAVs searches.

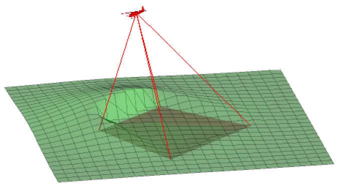

(a) prior belief (in green) of UAV 1 (in blue)

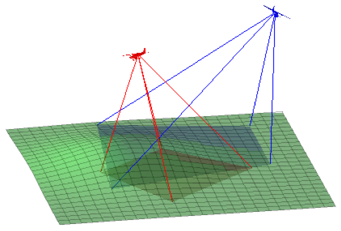

(b) observation of UAV 1

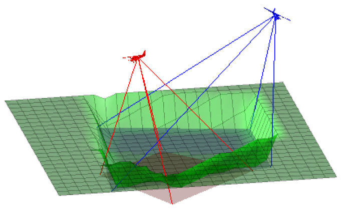

(c) posterior of UAV 1


(d) prior belief of UAV 2 (in red)


(e) observation of UAV 2


(f) posterior of UAV 2


(g) prior coordinated belief of UAV 1 and 2


(h) coordinated observation of UAV 1 and 2

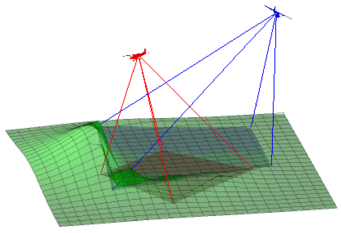

(i) coordinated posterior of UAV 1 and 2

FIGURE 3.4: Cross-coupling of utility functions. Given the same prior PDF, an observation of UAV 1, alone, gives a utility of 0.5192 (Figure 3.4(c)), while an observation by UAV 2 alone gives a utility of 0.2372 (Figure 3.4(f)). Finally, the utility of both UAVs is 0.5694 (Figure 3.4(i)).

Now, assuming conditional independence of the observations by the UAVs, we have:

$$P\left(x_{k+1}^{T}|z_{k+1},\ldots,z_1\right) = \frac{1}{C_2}P\left(x_{k+1}^{T}|z_k,\ldots,z_1\right)P\left(z_{k+1}|x_{k+1}^{T}\right) \tag{3.12}$$

Again, C_2 is a normalising constant. An illustration of the Bayesian update step is shown in Figure 3.4. Specifically, the figure depicts how the priors of two different UAVs 1 (blue in Figure 3.4) and 2 (red in Figure 3.4), shown, respectively, in Figures 3.4(a) and 3.4(d), are then updated with the two observations shown in Figures 3.4(b) and 3.4(e), to produce the posterior shown in Figures 3.4(c) and 3.4(f). The coordinated behaviour is shown in the remaining Figures of Figure 3.4. In this, case, by sharing their observations, the UAVs start with the same prior (Figure 3.4(g)), take and share their observations (Figure 3.4(h)) and finally compute the same posterior 3.4(i).

3.2.1.2 Prediction

The Bayesian prediction step is used to estimate the target's current state considering the target's motion model and the belief on its previous state, before incorporating the new observations. As with the Bayesian update equation, we adopted the prediction equation used in previous work (Bourgault et al., 2004; Furukawa et al., 2006). Adaptation to this problem gives:

$$P\left(x_{k+1}^{T}|z_k,\ldots,z_1\right) = \int P\left(x_{k+1}|x_k\right) P\left(x_k|z_k,\ldots,z_1\right) dx_k^t \tag{3.13}$$

where, $P\left(x_{k+1}|x_k\right)$ is the target probabilistic motion model described previously.

3.2.2 Coordination

In this section, we define the approach that was applied to the cooperative search problem. We first introduce the control space for the UAVs and the concept of receding horizon control (as discussed in Section 2.3.1). We then define the utility function and finally present the max-sum approach for explicit coordination. This approach is depicted as the decision module in Figure 3.2, and the three type of messages shared by the agents are the focus of the remainder of this section.

3.2.2.1 Control Space

In this work, we approximate the control space, which is commonly a continuous value, of each UAVs, by considering a finite set of trajectories that can be followed and from which the coordination strategies can choose. Clearly, this approximation reduces the precision of the overall decision process, since by using a discretisation we reduce the possible motion of a real UAV. However, this approach of discretising the action space of a robotic platform has been widely used in the robotics community, with a famous example being the online path planner on Stanley, the robot which won the DARPA Grand Challenge (Thrun et al., 2006). In this work, this set of pre-computed trajectories is calculated based on a set of nominal bank angles, in this case $\begin{bmatrix} -25^\circ & -8^\circ & 0^\circ & 8^\circ & 25^\circ \end{bmatrix}$. These were chosen as they give a good spread in the resulting control actions, which are illustrated in Figure 3.5. In particular, Figure 3.5 depicts three possible bank angles (in blue, green and red in the figure), and the corresponding footprints made by the UAVs.

Now, as we will see in more detail in Section 3.2.2.3, we will use this type of control space to estimate the future observations of a UAV. In few details, since each angle within the control space can be repeated for a finite number of time steps (as we will see in Section 3.2.2.2), it will constitute a specified path or trajectory the UAV will follow. Then,

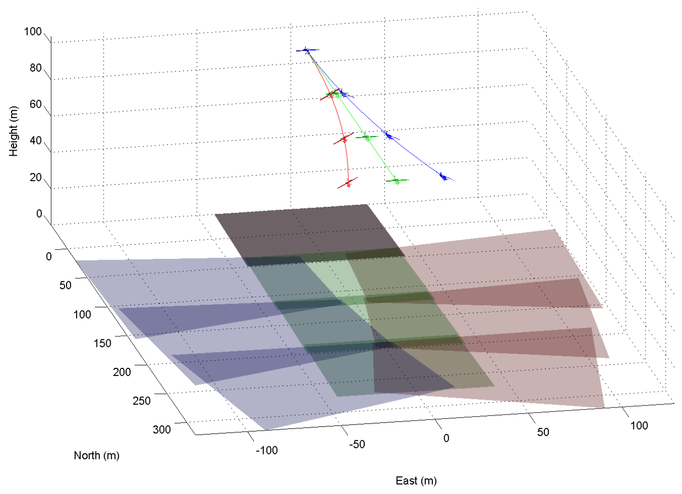


FIGURE 3.5: Control space of one UAV, and their associated sensor footprints.

within this path, the camera footprints, and thus the sensor model, can be estimated before taking the action.

The joint action space between two or more UAVs, then, is the set of all permutations of the individual control spaces of each UAV. With five members in the control space of one UAV, this means that two UAVs will have a joint control space of size 25, three UAVs will have a joint control space of size 125 and so on.

3.2.2.2 Receding Horizon Control

A number of factors mean that it is difficult or impossible to determine the control actions that the UAVs should take to find the target in the shortest time. These include:

- Imperfect models of the sensor characteristics and UAV motion and control.
- The dynamic nature of the environment, as manifested in the ever-changing PDF representing the belief of the state of the target. This means that in the time it takes to compute the optimal control action, the state of the world has changed, possibly rendering the computed action sub-optimal.
- The large search space, defined by the combinations of the control spaces of each UAV over the entire mission.

To compensate for these factors, receding horizon control is used to approximate the optimal solution. Receding horizon control was selected as it is a common technique used in the literature to give computationally tractable solutions given imperfect models of the world and a dynamic environment (as presented in Section 2.3.2). Put simply, receding horizon control chooses the action that is optimal over a given prediction horizon. This action is only executed for a length of time less than the prediction horizon, before a new optimal action is computed and the cycle is repeated.

3.2.2.3 Utility Function

The global utility function used in this work, which summarises up the performance of the UAV team, is the probability of detecting the target given all the observations made. The function was chosen because of its wide use in related research for representing the task of detecting a target (Bourgault et al., 2003, 2004) Over a prediction horizon of N steps and for a control action u, the utility function is defined as:

$$J\left(u, N\right) = P\left(\bar{D}_{1:k}\right) - P\left(\bar{D}_{1:k+N}\right) \tag{3.14}$$

where,

$$P(\bar{D}_{1:n}) = \prod_{m=1}^{n} P(z_m = \bar{D}|z_{m-1}, \ldots, z_1) \tag{3.15}$$

Intuitively, the utility $J(u, N)$, calculated at every time step k, measures how the joint action u will influence the current cumulative probability of detecting the target, if repeated over N time steps. In more detail, each agent m of the team simulates taking action $u_m \in u$ for N consecutive time steps, thus making N different observations. Now, all these observations are fused together with the ones made by the other agents and with the current value of $P(\bar{D}_{1:k})$ in order to calculate $P(\bar{D}_{1:k+N})$, i.e. the cumulative probability of detection after N time steps. Finally the utility $J(u, N)$ is calculated by subtracting $P(\bar{D}_{1:k})$ and $P(\bar{D}_{1:k+N})$, thus determining how the action u might influence the initial cumulative probability.

It can be seen that Equation 3.15 can be evaluated by calculating the cumulative product of the normalisation constants in the Bayesian update equation. To calculate utilities for each member of the control space, the observations that the UAV was predicted to make were fused into a copy of the PDF of the state of the target maintained by the UAV. As noted in the work which introduced this utility function, it attempts to maximise the increase in the cumulative probability of detection (Bourgault et al., 2004). This utility function was selected as it is already established in the search and track literature (Bourgault et al., 2004). Hence, the goal of the system is to find the joint control that maximises the global utility (for a detailed definition of how such joint paths are computed, refer to Section 3.2.2.1). However, the computation of the global utility for a joint control is not trivial, as the actions of one UAV may affect the utility of another UAV. This occurs when the sensor footprints of two UAVs overlap, as shown in Figure 3.4.

In order to apply the max-sum algorithm, the global utility function must be the sum of the individual contributions of each UAV in the team. To address this requirement, we decomposed the global utility function into the utilities of the individual UAVs using the concept of incremental utilities (Stranders, 2010). This approach consists of establishing an ordering of the agents in the team by assigning each agent a unique ID number. In the context of UAVs, this could be the tail number of UAV. The individual utility of UAV i is then defined as the incremental increase in the global utility function due to the predicted action of UAV i, considering the predicted actions of all UAVs j where $j < i$. It should be noted that the ordering chosen for the UAVs does not impact on the value of the calculated team utility. Moreover, it should be noted that this type of approach does not scale up to more than seven or eight agents (as shown in (Stranders et al., 2009)). The reason for this is that the agent with the highest index, needs to consider the predicted actions of *all* the other agents, and thus the computation becomes exponential in the number of agents. Other possible approaches exists, to address this issue, such as (Waldock et al., 2008), which we will discuss in more detail in Chapter 5. As an example, consider the utilities shown in Figure 3.4. The incremental utility of UAV (b) is calculated by taking into account the value of its own observations and those made by UAV (a). Specifically, the incremental utility of UAV (b) is calculated by subtracting the value in Figure 3.4 (a) from the value in Figure 3.4 (c), namely, $0.5694 - 0.5192 = 0.0502$.

3.2.2.4 The Max-Sum Approach

In order to have a thorough description of the max-sum algorithm we refer the reader to Section 2. Therefore, here, we focus our attention on the procedure that we use in order to adapt the max-sum algorithm to our setting.

In more detail, to characterise the factor graph for our specific problem, we define each agent m as controlling a variable x_m and a function node U_m. In particular, such function nodes will represent the agent's local utility function, or, in other words, the agent's local contribution to the control function $J(u, N)$. We chose here to use this approach because it shares many similarities with Stranders et al. (2009). However, it is important to note, that this is just one possible manner to represent the factor graph. Other approaches have been considered as well, where function nodes represent arbitrary constraints between the agents (Farinelli et al., 2008)[2] or where functions nodes represent target to track (Waldock et al., 2008). Once the factor graph is computed, the negotiation phase starts. During this phase, function and variable nodes each send a different type of message as defined by equations 2.2 and 2.3. Such negotiation takes place within this asynchronous message passing phase. The algorithm is guaranteed to converge to the optimal joint action if the factor graph has an acyclic structure. Otherwise, it finds an approximate solution (Farinelli et al., 2008). In this particular application, no assumptions can be made on the shape of the factor graph. Finally, to

[2]This is also the most general characterisation of a factor graph, the one we used in Chapter 2

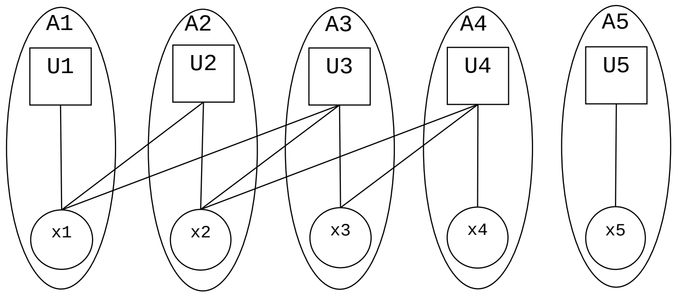


FIGURE 3.6: An example of a factor graph, involving 5 agents A_m. Here the predicted observations of UAV 5, are independent of those of all the other agents, while the predicted observations of UAV 4 are independent of those of UAV 1.

decide on a control action, the marginal function is calculated at each variable node, using equation 2.4, and the selected control, is calculated as the argument the maximise such equation.

Now, to apply the max-sum algorithm to our multi agent framework, we take inspiration from (Stranders et al., 2009), in which the max-sum algorithm has been successfully applied to a problem similar to ours. However, to adapt this method to the context of a team UAVs that coordinate in a realistic setting, a further level of communication, in which the agents communicate all their possible trajectories is required. We implement this additional level, by having all the UAVs share their predicted trajectories. Each variable x_m represents the possible trajectories that the m^{th} UAV can take, as defined in Section 3.2.2.1. Each function node U_m, then, represents the individual utility of the m^{th} UAV, as defined in Section 3.2.2.3. The edges of the factor graph are computed dynamically every time step, by building connections between function and variable nodes. In particular, each variable node x_m is always connected to the corresponding U_m and vice-versa, as the control action by the m^{th} UAV always affects its utility.

The remaining edges are computed considering the incremental utility, as defined in Section 3.2.2.3. In detail, the function node U_m, owned by UAV m, is connected to a subset of variables x_m, owned by a subset of UAVs m. Thus, following the definition of incremental utility, every UAV having a lower ID than m, might have its variable x_m connected U_m. Now, the connection between x_m and U_m, models the dependency between the predicted observations of both the agents, in particular when such predicted observations overlap, then the link between U_m and x_m is built. Whether two predicted observations overlap is determined by evaluating the utility of the two individual predicted observations, and the joint utility of both predicted observations together. If the two individual utilities sum to the joint utility, then the two actions are considered independent, and the variable node of the other UAV is not connected to the function node U_m. Otherwise, the variable node is connected to U_m. An example of a factor graph built following this procedure can be found in Figure 3.6.

We chose to use this utility-based method of determining whether predicted observations overlapped as it makes the source of the predicted observations anonymous, a property often required when building complex unmanned aircraft systems (UAS) (Cole, 2009). By this, we mean that the UAV receiving the predicted observations does not need to know about the sensing model of the UAV that transmitted it, as the sensing model is already encoded in the predicted observation.

It should be noted that the incremental nature of the individual contributions to the global utility function described in Section 3.2.2.3 tends to reduce the number of edges in this graph, as function nodes are only connected to variables nodes belonging to UAVs with a lower or equal ID. Additionally, a function node is only connected to a variable node if it depends on that variable, further sparsifying the graph. Indeed, literature shows that sparsity is fundamental in order to improve the computational efficiency of the max-sum algorithm (Stranders, 2010; Stranders et al., 2009).

3.3 Experiments

This section describes the results of simulations performed to test hypotheses on the effect of coordination on the performance of a team of UAVs. We tested such hypotheses within two different scenarios, with limited and unlimited communication range. We decided to use a common metric, widely used for measuring the performance of an information-based task (Grocholsky, 2002). Hence the performance metric was defined as the average time taken for a team of UAVs to obtain a 95% cumulative probability of detecting the target, conditioned on all the observations made by the UAVs up to that time. In other words, had there been a target in the search area, there would have been a 95% probability of detecting it in this time. At time k, this probability was calculated by $1 - P\left(\bar{D}_{1:k}\right)$. As noted previously, $P\left(\bar{D}_{1:k}\right)$ is the probability of not detecting the target up to time k, based on the observations by the team of UAVs up time k.

This section first presents a description of other approaches to coordination against which the max-sum algorithm was benchmarked in Section 3.3.1. Following this, the hypotheses themselves and the experimental methodology used to test these hypotheses are described in Section 3.3.2. Finally, the results of the experiments are presented in Section 3.3.3.

3.3.1 Benchmarks Algorithms

In this section, we outline the three approaches to coordination that the max-sum algorithm **(MS)** was benchmarked against. These approaches are classified into the three levels of coordination described in the previous chapter.

Non-Coordination (NC): In the non-coordinated approach, each UAV selects its control to optimise the utility function described in Section 3.2.2.3 over a given horizon, independently of the other UAVs. This optimisation occurs on the basis of different PDFs on the state of the target, since the UAVs also do not share observations.

Implicit Coordination (IC): In the implicitly coordinated approach, each UAV selects its control to optimise the utility function as above, however, in this case, the UAVs also communicate observations so that each UAV maintains the same belief of the state of the target as its neighbours, and makes decisions based on this shared belief. In this case, the implicit coordination arises because each UAV is making its decision based on a common belief of the state of the target.

Explicit Coordination with Best Response (BR): In the explicitly coordinated approach, the UAVs make a team decision based on both the common information, as well as the predicted observations communicated by other UAVs. The best response algorithm is an example of an explicit coordination algorithm that is the state of the art for the coordination of UAVs for search (as seen in Section 2.3.3). It is for this reason that this paper benchmarks the performance of the max-sum algorithm against the performance of the best response algorithm. The best response algorithm operates by having every UAV determine the best control action it can choose, given its belief, and given its knowledge about the control actions that the other UAVs of the team are going to take, based on what the other UAVs have previously communicated. Every UAV then broadcasts its new decision, and the cycle is repeated.

In the previous application of the best response algorithm for coordinating UAVs, two termination criterion were defined, a theoretical one, where the procedure was iterated until the optimal joint control action was found, and a practical one where the procedure iterated until the solution converged to a pre-defined threshold (Bourgault et al., 2004). In our case, to be able to compare this approach to the max-sum algorithm, we fixed the number of iterations that both the max-sum algorithm and the best response algorithms were allowed to go through before termination. For the simulations, this was fixed at six iterations[3]. Additionally, while the previous work optimised over a continuous control domain (Bourgault et al., 2004), the best response algorithm was only allowed to optimise over the discrete control domain available to the max-sum algorithm in this work.

[3]As shown in Stranders (2010), the number of iterations usually depends on the density of the factor graph, here we chose the number six, as it constitutes a reasonable estimate of the maximum distance, in terms of *edges of the factor graph* between two different agents

3.3.2 Methodology

This section describes the methodology that was used to test the following hypothesis in simulation:

Hypothesis: Teams of UAVs that explicitly coordinate using the max-sum algorithm will out-perform teams of the same size that explicitly coordinate by using the best response algorithm, which will in turn out-perform teams that are implicitly coordinating, which will out-perform teams that are not coordinated. Naturally, a shorter time taken to obtain a 95% cumulative probability of detecting the target meant that a team had performed better.

We tested this hypothesis in two different scenarios, with limited and unlimited communication. In both cases, in recognition of the possibility that the initial positions of the UAVs would influence the result, ten sets of random initial positions were generated for teams of different size, and a student's $t-$test was run in order to guarantee statistical significance.

Each team then flew a simulated search mission using each of the four types of coordination, starting from each of the ten initial positions. For each simulated mission, we considered a search area of 200×200m, and we specified a different altitude for each UAV, namely 60m, 80m, 100m, 120m and 140m. Moreover, we considered a prediction horizon of 6s and fixed the resolution of the grid to 200×200, that is, each cell represented a square of 1m^2.

For each simulation, we recorded the time to achieve a 95% confidence in detecting the target. After all 120 simulated missions were complete, the mission times for each team size/level of coordination combination was averaged across the ten initial positions. This gave average mission times for each team size/level of coordination combination.

In the unlimited communication case, simulations were run for teams of one, two and five UAVs and for each team size. A Student's t test with a 95% confidence interval was applied to determine if the differences between the average times for each type of coordination were statistically significant. Finally in the limited communication case, simulations were run only for teams of three UAVs, and to test the degradation of the algorithm's performance due to communication limitation, the simulations were run considering different communication ranges. The use of three UAVs only, is due to the fact that the main focus of these simulations were to test the robustness against communication failure and thus experiments for teams of one or two UAVs only would have been, respectively, useless and trivial. Furthermore, experiments with five UAVs would have required a higher computational effort, albeit providing results similar to the three UAVs case.

3.3.3 Results

We first present the results we obtained in the case of unlimited communications, and second the ones we obtained in the case of communications limited by range.

3.3.3.1 Unlimited Communications

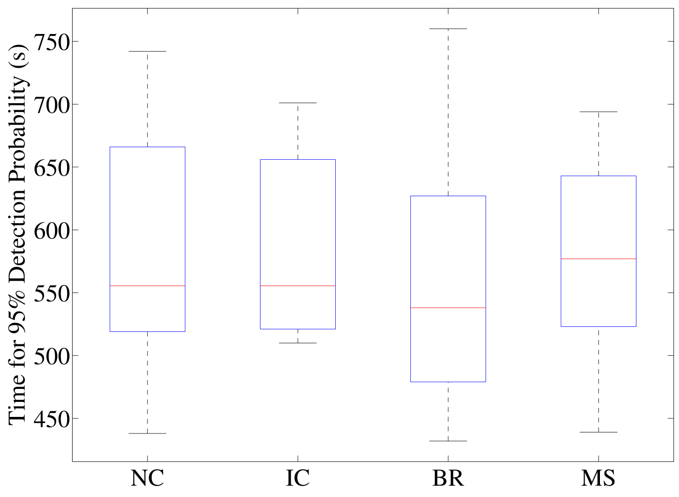
The results obtained from the simulated missions are illustrated in Figure 3.7. The differences between the times for the different types of coordination in the two and five UAV teams are evident. A Student's t test showed that the differences in the performance of the max-sum algorithm compared to the best response algorithm, implicit coordination and no coordination were statistically significant. Specifically, these results illustrated that for a team of two UAVs, explicit coordination with the max-sum algorithm reduced mission times by 7% compared with explicit coordination with the best response algorithm, by 17% compared with implicit coordination, and by 48% compared with no coordination. In a team of five UAVs, explicit coordination with the max-sum algorithm reduced mission times by 26% compared against explicit coordination with the best response algorithm, by 34% compared against implicit coordination, and 68% when compared against no coordination.

On the other hand, the differences that can be observed in the one UAV team are much smaller, and a Student's t test showed that these differences were statistically insignificant. This self-evident result verified that no level of coordination was unfairly advantaged or disadvantaged.

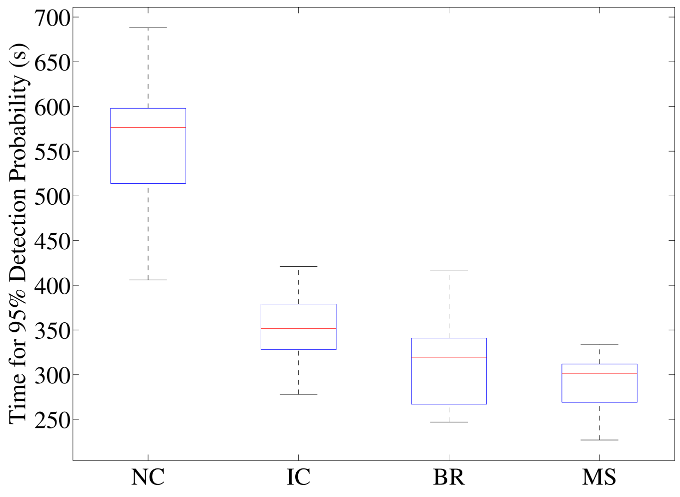
In terms of the size of the messages shared, the best response algorithm requires smaller messages than the max-sum algorithm. This is because once the predicted observations are communicated, the best response algorithm only communicates the index representing its best response at each iteration. On the other hand, the max-sum algorithm needs to communicate the variable to function and function to variable messages, which are vectors of length equal to the number of possible values each variable can take on, which is five in these simulations, there being five control actions in the control space.

3.3.3.2 Limited Communications

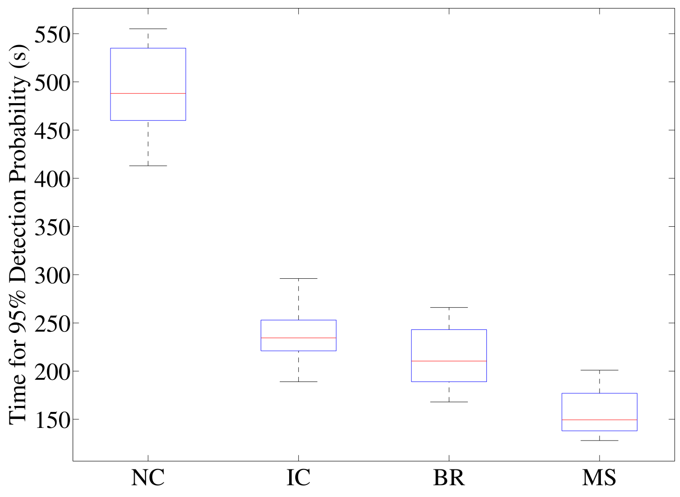
The results obtained from the simulated missions under limited communication range are illustrated in Figure 3.8. The max-sum approach clearly outperforms the other approaches and degrades gracefully for communication ranges down to approximately 375m. Indeed, in such cases, the performance of the max-sum algorithm is clearly better in comparison to the other approaches. Moreover the error bars in Figure 3.8, representing the standard error on the results obtained, show that such results are



(a) 1 UAV



(b) 2 UAVs



(c) 5 UAVs

FIGURE 3.7: Times to achieve a 95% cumulative probability of detection for one (a), two (b) and five (c) UAVs. The top and bottom of the central box in each box plot depicts the 25% and the 75% percentiles of the results, while the central mark represents the median.

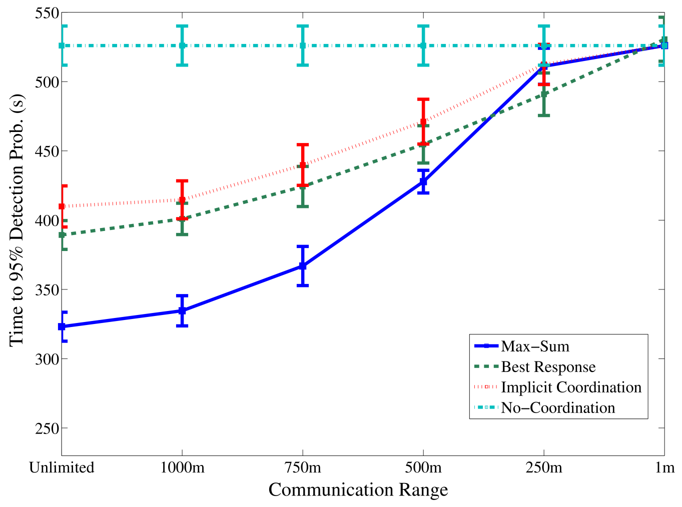


FIGURE 3.8: Time to achieve a 95% cumulative probability of detection for teams of three UAVs considering different communication ranges. The error bars in the chart represent the standard error obtained for each specific communication range.

clearly statistically significant. Below the 375m range, the difference of the performance of the different approaches becomes insignificant and a clear comparison can not be made anymore. Finally, as soon as communication becomes impossible (i.e near the 1m range), we can see that all these approaches degrade towards the non-coordinated level.

3.4 Summary

In this chapter, we presented our first contribution. In more detail, we introduced in Section 3.1 the framework we adopted in order to represent the team of UAVs (Section 3.1.1.2), in Section 3.2 we depicted our coordination approach. Specifically, we defined the Bayesian estimation model that we adopted in Section 3.2.1, we then introduced the tools that we used for coordinating the teams of UAVs, namely the control space and the receding horizon control, in Section 3.2.2. Finally, we concluded the section by describing the way we applied the max-sum algorithm to our specific problem.

In Section 3.3, we defined the experiments to test our approach. The simulation results showed that explicit coordination using the max-sum algorithm out-performed the best response algorithm, implicit coordination and no coordination. The simulation results further showed that explicit coordination using the best response algorithm outperformed implicit and no coordination. The simulation results finally showed that implicit coordination out-performed no coordination for both teams of 2 and 5 UAVs. From these observations, we can see that the hypothesis outlined previously is verified.

Now, the results obtained in this chapter show that max-sum is an effective algorithm in order to explicitly coordinate teams of robotic agents. Indeed, as we have mentioned at the beginning of the chapter, the algorithms addresses properly the requirements of timeliness, adaptiveness, robustness and autonomy. However, the requirement of scalability is only partially met, this is due to the fact that the incremental utility method that we presented in Section 3.2.2.3 cannot scale to more than a certain number of agents.

Moreover, in order to deploy such agents in real world domain, another key element is still missing: the ability of the algorithm to handle multiple objectives. Thus, in line with our research objectives, as stated in Chapter 1, the next chapter will build upon these results and describe the core contribution of this thesis, a novel decentralised coordination algorithm for addressing problems involving multiple objectives.

Chapter 4

Bounded Coordination over Multiple Objectives

Having discussed in the previous Chapter, the challenge of applying coordination mechanisms to a realistic domain, such as target search, we now address the challenge of developing coordination mechanisms able to address problems involving multiple objectives. More specifically, we address here two fundamental requirements, among those we defined in Chapter 1, namely the capability to address problems involving multiple objectives and to provide guarantees over the quality of the solutions recovered. To this end, we propose the bounded multi-objective max-sum algorithm (B-MOMS), a novel general decentralised algorithm able to solve coordination problems involving multiple objectives by providing guarantees on the recovered solutions. By doing this, we develop a fundamental milestone in order to solve information gathering problems involving multiple objectives, such as those presented in Chapter 2.

Thus, we address, here, the coordination problem from an abstract and theoretical perspective. In so doing, we aim to develop a general algorithm, in the sense that it allows teams of robotic agents to coordinate both in the information gathering domain and more broadly. As part of our future work, we will, then, consider the problem of applying the algorithm on the specific search and track problem.

In the remainder of this chapter we first define, in Section 4.1, the multi-objective distributed constraint optimisation problem (MO-DCOP), a novel general formalism that extends the standard DCOP model to the multi-objective case. Second, we characterise our algorithm in Section 4.2, by discussing in detail each of the three phases of our algorithm. Next, we derive a number of theoretical properties of our algorithm related to the type and the bound over the solutions that it computes in Section 4.3 and illustrate its behaviour by running a simple example in Section 4.4. We present some empirical evaluation of the performance of our algorithm in Section 4.5. Finally, Section 4.6 summarizes the chapter.

4.1 MO-DCOP Formalisation

We formalise the general problem by extending the DCOP framework, introduced in Chapter 2.3.4, to the setting of multiple objective functions. Hence, in contrast from Chapter 3, in this setting each agent $j \in (1, \ldots, M)$ controls only a variable x_j defined over a discrete domain D_j, while functions U_i^k $(i \in (1, \ldots N))$ represent constraints (related to a specific objective k) between sets of these variables.

More formally, we consider the multi-objective DCOP (MO-DCOP) problem, which involves the simultaneous optimisation of k DCOPs. Specifically, we consider the problem of maximising the following vector of objective functions:

$$\mathbf{U}(\mathbf{x}) = \left[U^1(\mathbf{x}), \ldots, U^k(\mathbf{x})\right]^T \tag{4.1}$$

Since each component of $U(\mathbf{x})$ is a DCOP, this global objective functions is decomposable into N factors, each of which is a multi-objective constraint function $\mathbf{U}_i$:

$$\mathbf{U}(\mathbf{x}) = \sum_{i=1}^{M} \mathbf{U}_i(\mathbf{x}_i)$$

where, again, each $\mathbf{x}_i \subseteq \mathbf{x}$ is the subset of variables representing the scope of multi-objective constraint function $\mathbf{U}_i$, which are defined as:

$$\mathbf{U}_i(\mathbf{x}_i) = \left[U_i^1(\mathbf{x}_i), \ldots, U_i^k(\mathbf{x}_i)\right]^T$$

Note that, for ease of exposition, we assume that all the local constraint functions U_i^k are defined over the same set of variables $\mathbf{x}_i$. However, this is not an essential requirement for the application of our algorithm.

Within this setting, the different solutions of a MO-DCOP are characterised using the solution concepts of Pareto optimality and non-dominance introduced in Section 2.4.1. Following these definitions, the solutions of a MO-DCOP are characterised as a set of Pareto optimal assignments PO corresponding to a set of non-dominated utilities vectors ND. Note that $\prod_{i=1}^{M} |D_i|$ 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.

Finally, we encode the MO-DCOP as a multi-objective factor graph by representing each variable x_i of the MO-DCOP as a variable node, while each function node now represents a vector function $\mathbf{U}_i$. The following example illustrates such a multi-objective factor graph.

Example 4.1. *Consider the factor graph in Figure 4.1, with three variables x_1, x_2, and x_3, which are controlled by agents $\mathcal{A}_1$, $\mathcal{A}_2$, and $\mathcal{A}_3$. There are three constraints between*

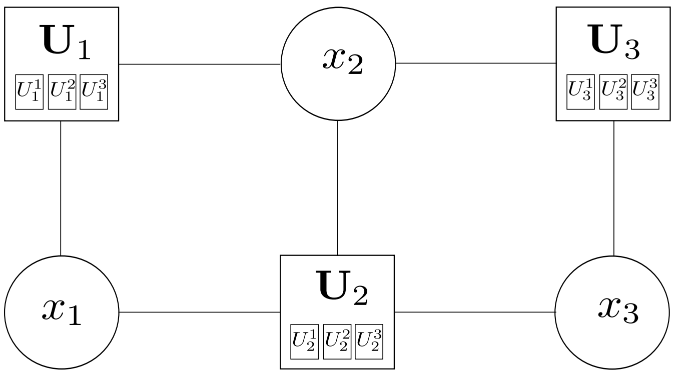


FIGURE 4.1: An example of a multi-objective factor graph, involving three variables, three objectives and three contraint functions.

the agents, represented by functions $\mathbf{U}_1$*,* $\mathbf{U}_2$*, and* $\mathbf{U}_3$*, each defined over three objectives* U^1*,* U^2*, and* U^3*.*

The example depicts the standard setting in which each factor contributes to each objective of the problem. This is not always the case, for instance, in problems such as search and track, the factors will represent tasks belonging to different objectives, such as targets to track or areas to search. A simple way to address this issue is to separate the different constraints U_i^k defined in each factor $\mathbf{U}_i$ into different factors. For instance, the factor $\mathbf{U}_1 = [U_1^1, U_1^2, U_1^3]$, depicted in Figure 4.1, can be separated into three factors $\mathbf{U}_1^1 = [U_1^1, 0, 0]$, $\mathbf{U}_1^2 = [0, U_1^2, 0]$ and $\mathbf{U}_1^3 = [0, 0, U_1^3]$ such that $\mathbf{U}_1 = \mathbf{U}_1^1 + \mathbf{U}_1^2 + \mathbf{U}_1^3$.

4.2 The B-MOMS Algorithm

Having described the MO-DCOP formalism, we now present the bounded multi-objective max-sum (B-MOMS) algorithm. B-MOMS extends the max-sum algorithm to compute solutions to MO-DCOPs. In detail, our algorithm proceeds in three phases:

- The *bounding* (B) phase, which extends the bounded max-sum algorithm discussed in Section 2.3.4.3, in order to provide quality guarantees. To achieve this, we generalise the maximum spanning tree algorithm to vector weights.
- The *max-sum* (MS) phase, during which the agents coordinate to find the Pareto optimal set of solutions to the cycle-free factor graph computed in the first phase. This requires a redefinition of the two key operations required by the max-sum algorithm (addition and marginal maximisation) for multiple objectives.
- The *value-propagation* (VP) phase, in which agents select a consistent variable assignment. This extends the standard VP phase used for the DPOP algorithm (as discussed in Chapter 2) to the case where multiple non-commensurable alternatives exist.

In what follows, we discuss each phase in more detail. Finally we present a complete example that illustrates the behaviour of our algorithm in Section 4.4.

4.2.1 The Bounding Phase

In order to bound the solutions of our algorithm, we propose a novel weighting approach that builds upon the existing bounded max-sum algorithm described in Section 2.3.4.3 by extending the edge weights w_{ij} to the multi-objective vector weights. The key issue is that the weighting approach used for max-sum, can be replicated for each of the different objectives of a multi-objective problem. To this end, we first compute the impact of each variable x_j in the scope of each local multi-objective function $\mathbf{U}_i$ over all k objectives:

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

Moreover, we define each scalar weight w_{ij}^o $(1 \leq o \leq k)$ as in the single-objective case:

$$w_{ij}^o = \max_{\mathbf{x}_i \setminus x_j} \left[\max_{x_j} U_i^o(\mathbf{x}_i) - \min_{x_j} U_i^o(\mathbf{x}_i)\right]$$

Since the problem of finding a maximum spanning tree is defined on instances with scalar edge weights, 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_{ij} to each vector $\mathbf{w}_{ij}$, which is proportional to the number of edge weights it dominates. More formally:

$$w_{ij} = -|\{\mathbf{w}_{mn} \mid \mathbf{w}_{mn} > \mathbf{w}_{ij}, (i,j) \neq (m,n)\}| \tag{4.2}$$

Thus, using this scalarisation, non-dominated weight vectors are assigned a value of 0, vectors dominated by a single elements are assigned a value of -1, and so on. With these scalar edge weights, the GHS algorithm can be used to compute a maximum spanning tree as discussed in Section 2.3.4.3.

4.2.2 The Max-Sum Phase

The second phase executes max-sum on the acyclic factor graph resulting from the previous phase. In order to apply max-sum to the multi-objective setting, however, the messages exchanged between functions and variables (Equations 2.2 and 2.3) need to be generalised to vectors of constraint functions.

Recall from Section 2.3.4 that, if the factor graph is acyclic, the messages r and q exchanged between U_i and x_j represent the maximum aggregate utility over the two components of the graph obtained by removing the dependency between U_i and x_j. The same holds in the multi-objective case. However, instead of $q_{j\to i}(x_j)$ and $r_{i\to j}(x_j)$ being scalar functions of x_j, these messages now map the domain of x_j to a *set* of utility vectors. To see why this is true, note that in the multi-objective domain, maximum utility is now defined in terms of the dominance relation from Definition 2.2. Since this relation induces a partial ordering, more than one such maximum might exist. To illustrate this, consider the following example:

Example 4.2. *Suppose the following message $r_{i\to j}(x_j)$ is sent by function $\mathbf{U}_i$ to variable x_j with domain $D_j = \{Red, Green, Blue\}$:*

$$r_{i\to j}(x_j) = \begin{cases} \{[0,0,0]\} & \textit{if } x_j = Red \\ \{[0,1,0]\} & \textit{if } x_j = Green \\ \{[-1,2,-1],[2,1,-1]\} & \textit{if } x_j = Blue \end{cases}$$

This message conveys the fact that, if x_j is assigned the value Red, the maximum possible utility obtained within the sub-graph connected to $\mathbf{U}_i$ after removing the dependency on x_j is equal to $[0,0,0]$. *Similarly, if $x_j = Blue$, there are two incomparable maxima (since neither dominates the other), namely* $[-1,2,-1]$ *and* $[2,1,-1]$.

To compute these messages, we generalise the two key mathematical operators required by max-sum—the addition of two vector functions (Equations 2.2 and 2.3) and the marginal maximisation with respect to a single variable ($\max_{\mathbf{x}_i\setminus x_j}$ in Equation 2.3)—to the multi-objective domain. Recall from Section 2.4.2.4, that similar operators were previously defined in the context of the multi-objective bucket elimination algorithm (Rollón, 2008). Thus, we redefine these operators here to compute the messages exchanged in the max-sum algorithm.

In more detail, to add two functions f and g defined over scope $\mathbf{x}$:

$$(f+g)(\mathbf{x}) = ND(\{\mathbf{v}+\mathbf{w}|\mathbf{v}\in f(\mathbf{x}); \mathbf{w}\in g(\mathbf{x})\})$$

where function $ND(A)$ filters out the dominated vectors from input set A. Using the + operator we can compute the sum of the messages r (which are univariate functions of x_j) in Equation 2.2, as well as the addition of multi-variate function $\mathbf{U}_j$ to the sum of univariate functions of *different* variables x_k in Equation 2.3.

The second operator, marginal maximisation ($\max_{\mathbf{x}_i\setminus x_j}$), takes as input a multi-variate vector function $f(\mathbf{x}_i)$ and outputs a function $f'(x_j)$:

$$f'(x_j = d) = ND\left(\bigcup_{\mathbf{d}\in D_{\mathbf{x}_i\setminus x_j}} f(\{\mathbf{x}_i\setminus x_j\} = \mathbf{d}, x_j = d)\right)$$

where $D_{\mathbf{x}_i\setminus x_j}$ is the Cartesian product of the domains of variables $\mathbf{x}_i \setminus x_j$.

At the end of the max-sum phase, each variable x_j computes its marginal function z_j (Equation 2.4) to obtain a set of Pareto optimal assignments A_j^*. Since there might be multiple optimal assignments, agents need to reach consensus on which global assignment to choose. Thus, in what follows, we define a value-propagation phase where the agents jointly choose a Pareto optimal solution among the ones computed by the max-sum phase.

4.2.3 The Value-Propagation Phase

The third and final phase, value-propagation, again operates on the cycle-free factor graph computed by the bounding approach. In this phase, variables and function nodes in this factor graph coordinate to select a consistent variable assignment. As we said at the beginning of this section, we develop this phase by building over the value-propagation phase used for the DPOP algorithm, that we discussed in Chapter 2 .

Now, at the end of the MS phase, each variable computes the set A_j^* of marginal Pareto optimal variable assignments for x_j, which is obtained by maximising over the marginal function z_j:

$$A_j^* = \arg\max_{x_j} z_j(x_j)$$

If, for any variable x_j, $|A_j^*| > 1$, then there exists more than a single global optimal assignment in the acyclic factor graph: $|\widetilde{\mathbf{PO}}| > 1$. If this is the case, we then propose a novel extension to the canonical value-propagation phase, in which a variable can select an assignment $a_j^* \in A_j^*$ at random, or one that satisfies some (logical) condition C. For example, a_j^* might be chosen such that objective 1 is maximised, subject to objective 2 being at least 4, or more formally, $a_j^* \in A_j^*$ and $\max z_j(a_j^*)_1$, subject to $z_j(a_j^*)_2 \geq 4$. To select an assignment that satisfies this condition, the value-propagation phase proceeds by passing messages between the variables and functions in the acyclic factor graph, and is thus fully decentralised. 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 Pareto optimal assignment a_r^* that satisfies C. The variable then sends value-propagation messages $(x_r = a_r^*)$ 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 $(x_j = a_j^*)$ from variable x_j, multi-objective constraint function $\mathbf{U}_i$ computes the set $\mathbf{A}^*$ of local Pareto optimal assignments for variables $\mathbf{x}_i \setminus x_j$, conditioned on $x_j = a_j^*$:

$$\mathbf{A}^* = \left\{ \mathbf{a} \mid \mathbf{a} \in D_{\mathbf{x}_i \setminus x_j}, g(\mathbf{a}) \in ND\Big(\bigcup_{\mathbf{a} \in D_{\mathbf{x}_i \setminus x_j}} g(\mathbf{a}) \Big) \right\}$$

where $g(a)$ is defined as:

$$g(\mathbf{a}) = \mathbf{U}_i(\{\mathbf{x}_i \setminus x_j\} = \mathbf{a}, x_j = a_j) + \sum_{k \in N(i) \setminus j} q_{k \to i}(a_k)$$

After computing set $\mathbf{A}$, value propagation selects a Pareto optimal assignment $\mathbf{a}^* \in \mathbf{A}$ that satisfies condition C and sends the message $(x_k = a_k^*)$ to every variable x_k $(k \neq j)$, where a_k^* is the element of $\mathbf{a}^*$ corresponding to x_k.

Variable nodes: For each non-root variable x_j, once it receives a message $(x_j = a_j^*)$ from a function $\mathbf{U}_i$, it sets its value to a_j^* and propagates the message $(x_j = a_j^*)$ to all the function nodes $\mathbf{U}_k$, $k \neq i$.

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.

4.3 Theoretical Analysis

We now discuss the theoretical properties of the B-MOMS algorithm.

4.3.1 Optimality of the Max-Sum Phase

The first property concerns the performance of the max-sum phase of the B-MOMS. Specifically, we show that the following theorem holds:

Theorem 4.1. *The max-sum phase (MS) computes the entire set of Pareto optimal solutions $\widetilde{\mathbf{PO}}$ of the acyclic factor graph that is obtained by pruning edges during the bounding (B) phase of the algorithm.*

Proof. The proof consists of two steps. Firstly, the valuation algebra defined by the $+$ and max operators discussed in Section 4.2.2, together with the co-domain of the global multi-objective constraint function $\mathbf{U}$ (Equation 4.1), is a commutative semi-ring (Rollón, 2008). Secondly, any GDL algorithm optimally solves problems whose valuation algebra is a commutative semi-ring, whenever the underlying constraint graph is acyclic (Aji and McEliece, 2000). Thus, since the second phase of B-MOMS is a GDL algorithm, the result holds. □

Now, while solutions $\widetilde{\mathbf{PO}}$ are Pareto optimal in the acyclic sub-graph, they are not necessarily (or likely) Pareto optimal in the original factor graph. However, using Theorem 4.1, we can derive bounds on the quality of these solutions in the original factor graph.

4.3.2 Bounded Approximation

To derive these bounds, we follow a procedure similar to that of the bounded max-sum algorithm (Section 2.3.4.3). First, we define vector $\mathbf{W} = [W^1, \ldots, W^k]$ as the sum of vector weights $\mathbf{w}_{ij}$ of the edges between U_j and x_i that were pruned in the bounding phase to obtain an acyclic graph (Section 4.2.1). Furthermore, to characterise the upper bound computed by B-MOMS, we first define the concept of *utopia point*:

Definition 4.2. Utopia Point (Marler and Arora, 2004): The utopia point $\mathbf{V}^*$ of a multi-objective optimisation problem characterised by objective function $\mathbf{U}(\mathbf{x}) = \left[U^1(\mathbf{x}), \ldots, U^k(\mathbf{x})\right]^T$ is given by:

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

Put differently, the utopia point is the vector of values resulting from optimising each k DCOPs independently. Clearly, given any Pareto optimal assignment $\mathbf{a}^* \in \mathbf{PO}$, $\mathbf{U}(\mathbf{a}^*) \leq \mathbf{V}^*$ holds for any MO-DCOP. Thus, the utopia point is an upper bound of the value of a Pareto optimal solution of a MO-DCOP. Given these concepts we can state the following theorem:

Theorem 4.3. *Given an arbitrary MO-DCOP, for any assignment* $\tilde{\mathbf{a}} \in \widetilde{\mathbf{PO}}$ *computed by B-MOMS, the following bound holds:*

$$\mathbf{U}(\tilde{\mathbf{a}}) + \mathbf{W} \geq \mathbf{V}^* \tag{4.3}$$

Proof. The theorem follows directly from the fact that we extend the bounded max-sum algorithm for single objective DCOPs to the case of multi-objective problems. 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:

$$U^o(\tilde{\mathbf{a}}) + W^o \geq \max_{\mathbf{x}} U^o(\mathbf{x})$$

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 2.6.

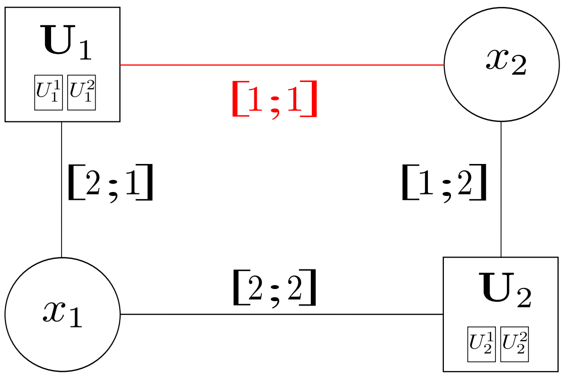


FIGURE 4.2: A factor graph, involving two variables x_1 and x_2 and two vector functions $\mathbf{U}_1$ and $\mathbf{U}_2$. The red edge is the one that is going to be pruned during the bounding phase.

4.3.3 Complexity

Finally, we derive the computation and communication complexity of B-MOMS using properties inherited from the max-sum algorithm. Now, since B-MOMS exploits the factorisability of the problems it is solving, the scope of each constraint function $\mathbf{U}_i(\mathbf{x}_i)$ contains only the variables on which the constraint is defined. Therefore, computing message $r_{i \to j}$ from function $\mathbf{U}_i$ to variable x_j (Equation 2.3) requires $O(|D_{max}|^{|\mathbf{x}_i|})$ evaluations of function $\mathbf{U}_i$, where D_{max} is the largest domain among variables $\mathbf{x}$. Hence, the computation is exponential *only* in the number of variables in the scope of $\mathbf{U}_i$, not the total number of variables.

Furthermore, since in the worst case every variable assignment of $\mathbf{x}$ is Pareto optimal, after a certain (but finite) number have been exchanged, these messages contain $O(k \times |D_{max}|^{M+1})$ values: $|D_{max}|$ sets of vectors (one for each value in the variable's domain), each containing at most D_{max}^M non-dominated vectors of size k.

4.4 Example

We present in this section an example to illustrate the behaviour of B-MOMS. We consider the factor graph in Figure 4.2, where two agents x_1 and x_2 coordinate over the vector function $\mathbf{U}(x_1, x_2) = \mathbf{U}_1(x_1, x_2) + \mathbf{U}_2(x_1, x_2)$, and illustrate the three different phases of our algorithm.

Example 4.3. *We consider a bi-objective function as defined in Table 4.1. In such a case the set of non-dominated vectors is defined as:*

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

x_1	x_2	$\mathbf{U}_1(x_1,x_2)$	$\mathbf{U}_2(x_1,x_2)$	$\mathbf{U}(x_1,x_2)$
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	$[2,1]$	$[0,1]$	$[2,2]$

TABLE 4.1: The bi-objective function corresponding to the graph in Figure 4.2.

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

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

Moreover, the utopia point, follows from Definition 4.3.2:

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

We now characterise the three different phases of our algorithm:

Bounding phase: *In the first phase the two factors* $\mathbf{U}_1$ *and* $\mathbf{U}_2$ *compute their vector weights as detailed in Section 4.2.1. Specifically,* $\mathbf{U}_1$ *computes:*

$$w_{11}^1 = \max_{x_2}[\max_{x_1} U_1^1(x_1,x_2) - \min_{x_1} U_1^1(x_1,x_2)] = 2$$
$$w_{11}^2 = \max_{x_2}[\max_{x_1} U_1^2(x_1,x_2) - \min_{x_1} U_1^2(x_1,x_2)] = 1$$

where w_{11}^1 *and* w_{11}^2 *are the two weights related to the edge between* $\mathbf{U}_1$ *and* x_1*. Moreover, the factor computes:*

$$w_{12}^1 = \max_{x_1}[\max_{x_2} U_1^1(x_1,x_2) - \min_{x_2} U_1^1(x_1,x_2)] = 1$$
$$w_{12}^2 = \max_{x_1}[\max_{x_2} U_1^2(x_1,x_2) - \min_{x_2} U_1^2(x_1,x_2)] = 1$$

where w_{12}^1 *and* w_{12}^2 *are the two weights related to the edge between* $\mathbf{U}_1$ *and* x_2*. In a similar fashion* $\mathbf{U}_2$ *computes:*

$$w_{21}^1 = \max_{x_2}[\max_{x_1} U_2^1(x_1,x_2) - \min_{x_1} U_2^1(x_1,x_2)] = 2$$
$$w_{21}^2 = \max_{x_2}[\max_{x_1} U_2^2(x_1,x_2) - \min_{x_1} U_2^2(x_1,x_2)] = 2$$

where w_{21}^1 *and* w_{21}^2 *are the two weights related to the edge between* $\mathbf{U}_2$ *and* x_1*. Finally, the factor computes:*

$$w_{22}^1 = \max_{x_1}[\max_{x_2} U_2^1(x_1,x_2) - \min_{x_2} U_2^1(x_1,x_2)] = 1$$
$$w_{22}^2 = \max_{x_1}[\max_{x_2} U_2^2(x_1,x_2) - \min_{x_2} U_2^2(x_1,x_2)] = 2$$

x_1	x_2	$\min_{x_2^c} \mathbf{U}_1(x_1, x_2^c)$	$\mathbf{U}_2(x_1, x_2)$	$\mathbf{U}(x_1, x_2)$
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 4.2: The resulting bi-objective function after pruning the factor graph in Figure 4.2. Here x_2^c is the variable whose corresponding edge has been pruned from the factor graph.

where w_{22}^1 and w_{22}^2 are the two weights related to the edge between $\mathbf{U}_2$ and x_2. The resulting weighted factor graph is shown in Figure 4.2. 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 4.2.1:

$$\begin{aligned} w_{11} &= -3 \\ w_{12} &= -1 \\ w_{21} &= -1 \\ w_{22} &= 0 \end{aligned} \tag{4.4}$$

Thus, since w_{11} is the smallest weight on the novel weighted factor graph, it is pruned from the graph by the GHS algorithm. The set of values, corresponding to the cycle free factor graph is shown in Table 4.2. 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 4.2 constitutes the input of the second phase of the algorithm, which follows.

The Max-Sum phase: *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 $\mathbf{U}_1$ 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:*

$q_{2\to2}(x_2) =$

x_2	$\mathbf{U}$
0	$[0,0]$
1	$[0,0]$

$r_{1\to1}(x_1) =$

x_1	$\mathbf{U}$
0	$[2,0],[0,1]$
1	$[1,0]$

Here x_2 sends a "zero" message because it has not received any information, while $\mathbf{U}_1$ computes the multi-objective marginal function of variable x_1, as detailed in Section 4.2. At time step $t = 1$, x_1 and $\mathbf{U}_2$ receive the messages and compute the next two messages:

$$r_{2\to 1}(x_1) = \begin{array}{c|c|} x_1 & \mathbf{U} \\ \hline 0 & [3,2] \\ 1 & [2,4] \\ \hline \end{array} \qquad q_{1\to 2}(x_1) = \begin{array}{c|c|} x_1 & \mathbf{U} \\ \hline 0 & [2,0],[0,1] \\ 1 & [1,0] \\ \hline \end{array}$$

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 $\mathbf{U}_1$ (i.e. since the original edge has been pruned from the graph). Thus, x_1 and function $\mathbf{U}_2$ compute the following messages:

$$q_{1\to 1}(x_1) = \begin{array}{c|c|} x_1 & \mathbf{U} \\ \hline 0 & [3,2] \\ 1 & [2,4] \\ \hline \end{array} \qquad r_{2\to 2}(x_2) = \begin{array}{c|c|} x_2 & \mathbf{U} \\ \hline 0 & [5,2],[3,3] \\ 1 & [3,3] \\ \hline \end{array}$$

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

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

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

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

The value-propagation phase: *At time step 3, variable x_2, as the pre-determined root of the graph, selects a specific assignment*[1] *and sends a VP message containing the selected value $\tilde{a}_2 = 0$ to $\mathbf{U}_2$. At time step 4, $\mathbf{U}_2$ receives the message, calculates the set of Pareto optimal solutions for which $x_2 = 0$, that is $\tilde{\mathbf{A}} = \{[x_1 = 0, x_2 = 0], [x_1 = 1, x_2 = 0]\}$, selects one of the different alternatives*[2] *and sends a new VP message to x_1 containing $\tilde{a}_1 = 1$. Finally, at the time steps 5 and 6, x_1 receives the message, forwards it to $\mathbf{U}_1$, and the computation terminates.*

[1]To keep the exposition as clear as possible, we assume that the selection criterion is to pick a random value.

[2]Again, we assume a random selection.

Thus, the algorithm recovers the solution $\tilde{\mathbf{a}} = \{1, 0\}$, which is, as can be seen in Table 4.1, one of the Pareto optimal solutions of the problem, in this specific case.

4.5 Empirical Evaluation

We present in this section some empirical evaluation in order to verify the performance of the B-MOMS. To this end, in this section we benchmark the performance of B-MOMS against an optimal algorithm. 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.

4.5.1 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 the DCOP literature (Farinelli et al., 2008). In this multi-objective graph colouring problem, 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' goals is to maximise the following multi-objective function:

$$\mathbf{U}(\mathbf{x}) = \Big[U^1(\mathbf{x}), U^2(\mathbf{x}), U^3(\mathbf{x})\Big]^T \tag{4.5}$$

where U^1, U^2, U^3 are the sum of bi-variate constraint functions $U_i^1(x_j, x_k)$, $U_i^2(x_j, x_k)$, $U_i^3(x_j, x_k)$ that exist among the variables $\mathbf{x}$. These three types of constraint functions are defined as follows:

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

$$U_i^1(x_j, x_k) = \begin{cases} 0 & x_j \neq x_k \\ -1 & x_j = x_k \end{cases} \tag{4.6}$$

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

$$U_i^2(x_j, x_k) = \begin{cases} 0 & \text{if } j < k \text{ and } x_j < x_k \\ -1 & \text{otherwise} \end{cases} \tag{4.7}$$

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_j and x_k, the distance of the colours between the two variables should equal one:

$$U_i^3(x_j, x_k) = \begin{cases} 0 & \text{if } |x_j - x_k| = 1 \\ -1 & \text{otherwise} \end{cases} \tag{4.8}$$

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 $\mathbf{U}_i(x_j, x_k) = [U_i^1(x_j, x_k), U_i^2(x_j, x_k), U_i^3(x_j, x_k)]^T$ where x_j and x_k are the variables corresponding to vertices v and v'.

4.5.2 Experimental Setup

We analyse the performance of B-MOMS 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 $\delta = \{0.0, 0.1, \ldots, 1.0\}$. Moreover, for each combination of values for M and δ we ran B-MOMS 120 times to achieve statistical significance. We measure the performance of B-MOMS using the following four metrics:

1. The runtime (in milliseconds) required by B-MOMS to compute the set of solutions.
2. The average approximation ratio of the solutions $\widetilde{\mathbf{PO}}$ computed by B-MOMS. More specifically, we defined this as the norm of the approximation ratio vector $||\boldsymbol{\rho}||$ defined in Section 4.3.2.
3. The minimum Euclidean distance between solutions $\mathbf{a} \in \widetilde{\mathbf{PO}}$ computed by B-MOMS and an optimal solution $\mathbf{a}^* \in \mathbf{PO}$. More formally, we calculate this distance as follows:
$$d(\mathbf{a}) = \left\| \min_{\mathbf{a}^* \in PO} \left[\mathbf{U}(\mathbf{a}) - \mathbf{U}(\mathbf{a}^*) \right] \right\|$$
4. Finally, we measure the fraction Pareto optimal solutions found by B-MOMS:
$$n_{PO} = \frac{|\widetilde{\mathbf{PO}} \cap \mathbf{PO}|}{|\mathbf{PO}|}$$

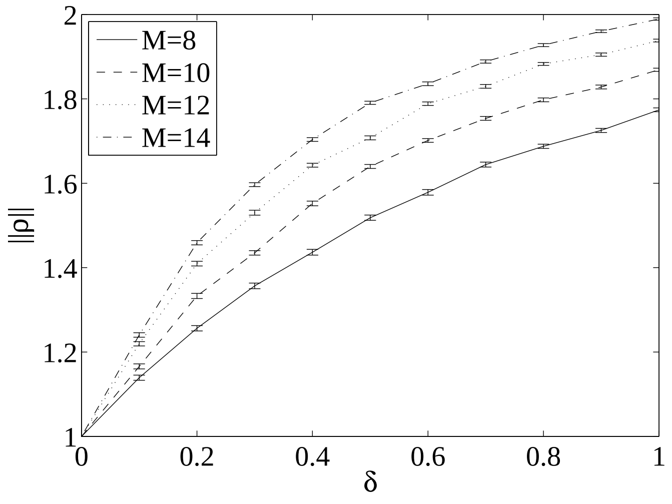
Note that metrics 2 and 3 aggregate the objectives by using the notions of norm and Euclidean distance, which implies commensurability of the objectives. However, these metrics have been widely used in multi-objective literature (Veldhuizen and Lamont, 1998). Moreover, metrics 3 and 4 require the availability of the set of Pareto optimal solutions **PO**. To compute these, we used a centralised 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 we were unable to run a sufficient number of experiments to obtain statistically significant results. However, we do report metrics 1 and 2 for M up to 100.

4.5.3 Results

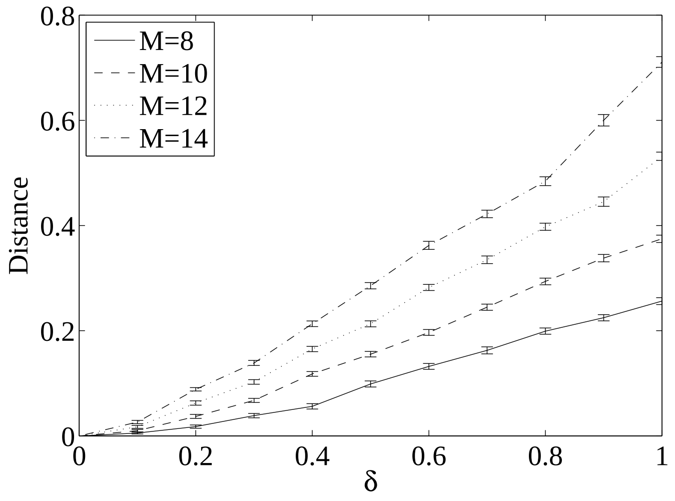
For $M \leq 14$, the results are shown in Figure 4.3. First of all, all three plots confirm that the algorithm is optimal for acyclic graphs ($\delta = 0$), as proved by Theorem 4.1. Moreover, by increasing the number of constraints of the problem (i.e. by increasing δ), we can observe that the performance 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, B-MOMS recovers roughly 50% of the optimal solutions for $M = 14$.

Figures 4.4(a) and 4.4(b) report the approximation ratio and the runtime for larger problem instances. Specifically, Figure 4.4(a) clearly shows that, even for large instances, the approximation ratio again 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 4.4(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[3], 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 B-MOMS well within the realm of the limited computational capacities of embedded agents found in many real-life applications.

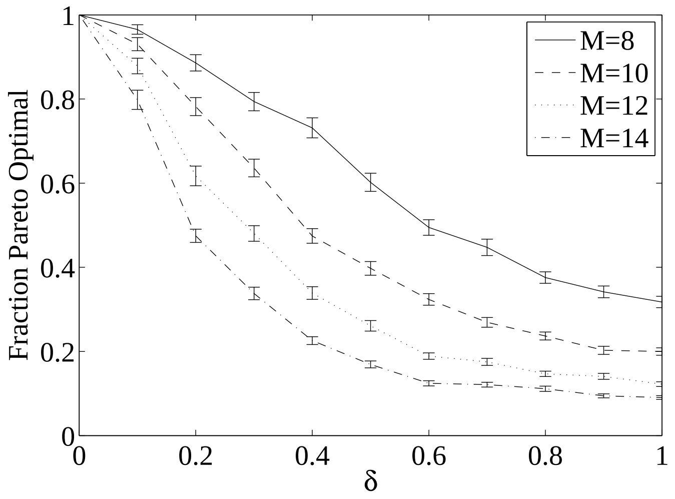
[3]This is partially due to our implementation of the value-propagation phase, which for the purpose of these experiments, recovers an exponential number of (approximately) Pareto optimal solutions, instead of just a single one.



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

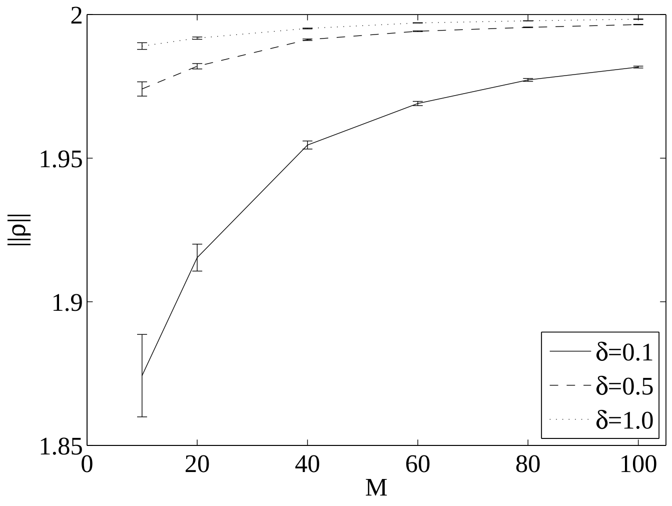


(b) Minimum distance of computed solutions to a Pareto optimal solution

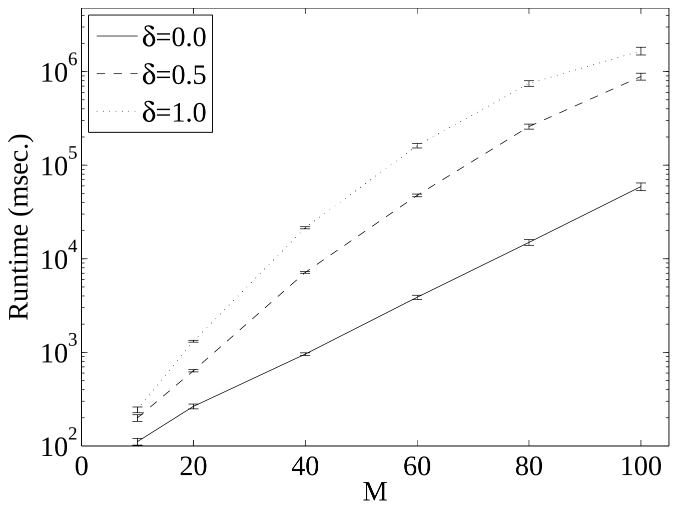


(c) Fraction n_{PO} of Pareto optimal solutions found

FIGURE 4.3: Empirical results for $M \leq 14$. Errorbars indicate the standard error of the mean.



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



(b) The runtime for varying number of vertices

FIGURE 4.4: Empirical results for $10 \leq M \leq 100$. Errorbars indicate the standard error of the mean.

4.6 Summary

We proposed in this chapter the second contribution of this thesis, the bounded multi-objective max-sum algorithm (B-MOMS), the first decentralised coordination algorithm for multi-objective optimisation problems. B-MOMS extends the bounded max-sum algorithm for decentralised coordination (Rogers et al., 2011) to compute bounded approximate solutions to multi-objective DCOPs (MO-DCOPs), a novel extension to the DCOP framework that we presented in Section 4.1.

Next, we introduced each of the three phases characterising our algorithm. In Section 4.2.1, we introduced the first phase of our algorithm. This phase extends the bounded max-sum algorithm to compute a cycle-free sub-graph of the multi-objective factor graph, which involves a generalisation of the maximum spanning tree problem to vector weights. In Section 4.2.2, we characterised the second phase of B-MOMS, in which we generalised the key mathematical operators required by max-sum to optimally solve the multi-objective problem encoded in the former cycle-free factor graph. In Section 4.2.3, we discussed the third and final phase which is required whenever there are multiple Pareto optimal assignments to the cycle-free problem. In Section 4.3, we proved the optimality of B-MOMS in acyclic constraint graphs, and derived bounds on the approximation ratio.

Next, we benchmarked B-MOMS against an optimal centralised algorithm on a multi-objective extension of the graph colouring problem. We discussed the results we obtained in Section 4.5.3. Such results show that the algorithm performs well in terms of the approximations that it computes, even for complex problem instances (defined by three objectives) and thus that it satisfies the multi-objective and quality guarantees requirements mentioned in Section 1. Moreover the algorithm performs well in terms of the time required to compute the set of Pareto optimal solutions, and thus it properly meet as well the timeliness requirement mentioned in Chapter 1.

Chapter 5

Conclusions and Future Work

We have presented two novel decentralised algorithms to coordinate teams of robotic agents that need to be deployed in real world applications, such as information gathering tasks. The first approach consists in a coordination approach for teams of UAVs deployed within an unknown environment to search for a target, while the second consists in a general decentralised algorithm for coordinating multiple agents over multiple objectives.

We present in this chapter an analysis of the approaches we developed. In more detail, we summarise our work in Section 5.1, and finally we present the future work in Section 5.2.

5.1 Conclusions

The work in this thesis has been divided in four different chapters. In Chapter 1, we introduced the field of work, in Chapter 2 we analysed the literature relevant to this work, while in the Chapters 3 and 4 we presented our contributions.

In more detail, in Chapter 1 we discussed the challenges and the issues that arise when coordinating teams of robotic agents for information gathering tasks. In this context, we outlined the different requirements— timeliness, robustness, scalability, adaptiveness, autonomy, the ability to handle multiple objective and to provide quality guarantees— that these coordination mechanisms should fulfil in order to constitute an effective solution technique. These are the criteria against which our algorithms will be evaluated.

In Chapter 2, we analysed the academic literature relating to our specific area of work. We focused our attention on one specific information gathering problem, target search, in which a team of robotic agents explore an outdoor environment looking for a specific target. In this setting, we showed how unmanned aerial vehicles are the best possible platform to employ, because they are able to gather information in a more effective way. Moreover, we further discussed how in many real world domains, the target search problem is extended in order to incorporate the task of tracking those targets that have already been detected, in order to localise, with high precision their position. Next, we examined the academic literature on coordination algorithms. We initially focused on existing decentralised coordination approaches. Such approaches are commonly divided in three different levels depending on the type of information shared between the agents. In more detail, we first discussed non-coordinated approaches in which each member of a team of agents behaves independently, we then presented implicit coordination approaches, where these agents build a common world view about the state of the target by sharing their observations, and finally we depicted explicit coordination approaches where the agents share, again, their observations and then make a joint decision about what to do next. Unfortunately, none of these approaches meet all our requirements. More specifically, some have exponential features, while others cannot guarantee an adequate level of performance. In response to this, we introduced the max-sum algorithm, a specific type of approximate message-passing algorithm, that has been shown to meet many of the requirements presented in Chapter 1 and was successfully used for problems similar to ours. One specific requirement that is not met by any current coordination approaches is that of handling multiple objectives. For this reason, we concluded Chapter 2 by presenting literature on multi objective optimisation a specific field in both Artificial Intelligence and Operational Research that studies problem involving the simultaneous optimisation of multiple incommensurate objectives.

In Chapter 3 we focused our attention on the problem of coordinating teams of UAVs for target search. Within this domain, we presented our first contribution. Specifically, we applied the max-sum algorithm to coordinate a team of UAVs to search for a dynamic target. We initially formalised the problem by considering a common setting, that has been widely used in literature (Bourgault et al., 2003, 2004). We then benchmarked the max-sum algorithm against three approaches representing the level of coordination defined above, namely the best response algorithm, representing the state of the art in the coordination of UAVs for search, an implicitly coordinated and a non coordinated approach, which we used as a lower bound on the performance of our algorithms. To compare the performance of the different approaches, we measured the average time taken for a team of UAVs to obtain a given confidence of detecting the target within an unlimited coordination scenario, then we compared the same approaches when communication becomes range limited. By doing this, we showed that coordination with the max-sum algorithm out-performed the best response algorithm by 26%, an implicitly coordinated approach by 34% and a non-coordinated approach by 68% in the case of five UAVs, and by 7%, 17% and 48% respectively for teams of two UAVs in the case of unlimited communications. We finally showed how the performance of the max-sum algorithm degrades gracefully when communication is limited, also outperforming all the other approaches. Thus, these results indicates that the max-sum algorithm has the potential to be applied in complex systems operating in dynamic environments.

Next, we addressed the general problem of coordinating with multiple objectives. Thus, in Chapter 4, we proposed the bounded multi-objective max-sum algorithm (B-MOMS), the first decentralised coordination algorithm for multi-objective optimisation problems. B-MOMS extends the bounded max-sum algorithm for decentralised coordination to compute bounded approximate solutions to multi-objective DCOPs (MO-DCOPs), a novel extension to the DCOP framework. It consists of three phases. The first phase extends the bounded max-sum algorithm (Rogers et al., 2011), to compute a cycle-free sub-graph of the multi-objective factor graph, which involves a generalisation of the maximum spanning tree problem to vector weights. The second phase generalises max-sum to optimally solve the multi-objective problem encoded in this cycle-free factor graph. Since there might be multiple Pareto optimal assignments to the cycle-free problem, the third and final phase enables agents to reach consensus on which global assignment to choose. We then proved the optimality of B-MOMS in acyclic constraint graphs, and derived bounds on the approximation ratio. Furthermore, we benchmarked B-MOMS against an optimal centralised algorithm on a multi-objective extension of the graph colouring problem and demonstrated that the approximation ratio never exceeds 2, and is typically less than 1.5 for graphs in which dependencies exist between 20% of all pairs of agents. Moreover, the runtime required by B-MOMS never exceeds 30 minutes, even for maximally constrained graphs with a hundred agents, positioning it well within the confines of real-life applications.

5.2 Future Work

The approaches presented so far fulfil only part of the requirements described in Chapter 1 and a number of some issues still need to be addressed in order to develop effective techniques. In particular, two of the challenges presented in Chapter 1 still need to be taken into account. Moreover, if we relate our approaches to the requirements presented in Chapter 1, we identify a set of requirements that still needs to be considered or that necessitates some further improvements in order to be properly satisfied.

It is important to note that since both our approaches build on the max-sum algorithm, they, consequently, inherit most of the main features of the algorithm. Thus, both our approaches are autonomous and robust to component and communication failure (Farinelli et al., 2008). Moreover, since max-sum is known to require little computation and memory space in order to compute solutions, our approaches clearly meet the requirement of timeliness as well. We address, then, the issue of adaptiveness in the approach that we present in Chapter 3. Indeed, by building an explicit coordination approach incorporating both max-sum and decentralised data fusion, we enable the agents to constantly update their beliefs and, therefore, their joint decision process depending on the dynamism of the environment. Next, we address two further requirements, the ability to provide quality guarantees and the ability to handle multiple objective in the algorithm we present in Chapter 4. As we have shown in Chapter 4, the bounded multi-objective max-sum algorithm is able to handle decentralised optimisation problems, while simultaneously providing bounds on the solution it has recovered.

Now, in order to build a coordination approach that is able to meet all the above requirements entirely, we will address, in the remainder of this PhD, the following challenges:

1. We will investigate the possibility of developing an explicit coordination approach, such as the one we presented in Chapter 3, but based on the B-MOMS algorithm. The resulting algorithm, would then be adaptive to the behaviour of the targets, because the agents consistently update their beliefs through data fusion, while it would also be able to handle multiple objectives and to provide quality guarantees over the solutions it recovers. Clearly, our aim will be to apply this technique to the multi-objective extension of target-search that we presented in Chapter 2, the search and track problem.

2. Next, our future work will aim to investigate the scalability requirement, which our explicit coordination approach still lacks. This is not due to the max-sum algorithm, as it has been shown to scale well if the factor graph representation is carefully chosen (Farinelli et al., 2008). The main challenge relates to the mechanism used to calculate the dependencies between the agents' predicted observations (as explained in Sections 3.2.2.3 and 3.2.2.4), which becomes rapidly unfeasible as the number of agents grows. To address this shortcoming, we will investigate other means of representing the dependencies between the agents, which would allow a higher degree of scalability. One interesting starting point could be to represent the search problem as a task assignment problem, as presented in Waldock et al. (2008) for the case of target tracking.

3. As mentioned in Chapter 1, our coordination mechanisms should be able to address the issue of *uncertainty* over the global utility function. Indeed, this uncertainty is *endemic* in real world domains and current state-of-the-art coordination mechanisms fall short in providing an effective countermeasure against it. In more detail, considering a setting where the agents have to coordinate their actions, while being uncertain about the values of their utility function, all the current state-of-the-art coordination mechanisms fall short in providing any type of quality guarantees on the type of solution that they recover. In this sense, our challenge will be to develop an algorithm able to guarantee a precise bound on the type of solutions that it retrieves.

4. Finally, as we mentioned, again, in Chapter 1, our coordination mechanisms need to be tested on field experiments on real UAVs. To date, we only run off-line simulations of our approaches, therefore we need to test our algorithms on the field to properly confirm their effectiveness. This second challenge, further includes other

interesting aspects. Indeed, running field experiments will help us understand in what manner we can ameliorate our approaches in order improve their usefulness in real world operations. Thus, it will allow us to fulfil the main aim of this PhD: developing coordination techniques able to enhance the deployment of teams of autonomous robotic agents in real world operations.

Currently, we are investigating the first of these two challenges. In particular, we are investigating the possibility to develop a novel decentralised coordination algorithm, that is able to provide quality guarantees over the solution it recovers, when the global utility function over which the agents are coordinating is uncertain. The focal point of this work will be to understand how to increase the computational complexity of the algorithm in order to guarantee a high quality bound over the solutions it computes. Next, we will address the challenge of testing our approach on field experiments, by spending part of this PhD visiting the Australian Centre for Field Robotics (ACFR) at the University of Sydney. The main purpose of the experience will be to test the approach on real unmanned aerial vehicles, therefore, showing the real effectiveness of the approach and, finally, improving our current knowledge about robotic platforms such as UAVs.

Figure 5.1 depicts a Gantt chart in which the schedule of work that we intend to do in the remaining months of this PhD is shown.

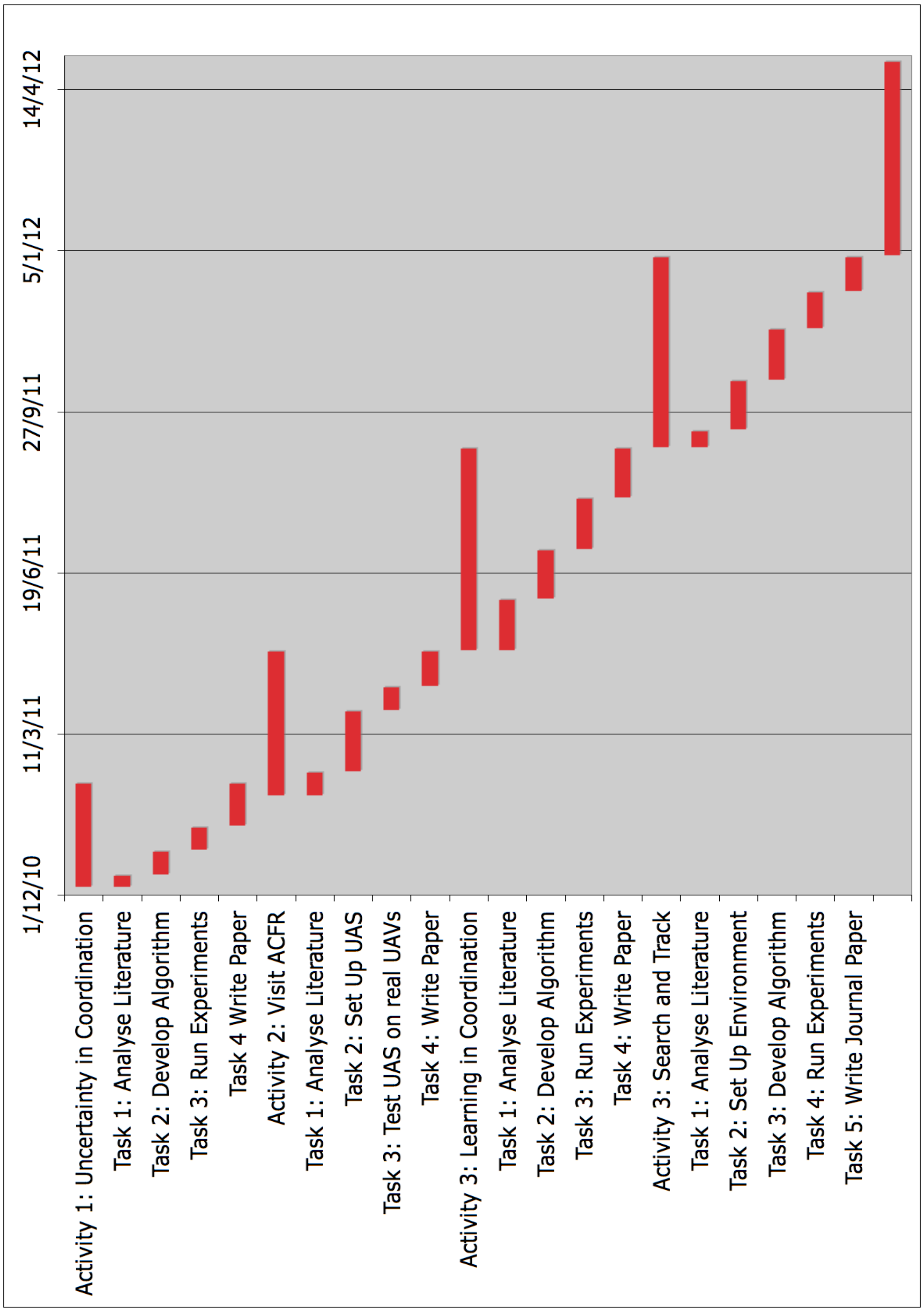


Figure 5.1: Gantt Chart

Bibliography

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

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*, pages 756–761, 2003. Maui, USA.

S. Bistarelli, U. Montanari, and F. Rossi. Semiring-based constraint satisfaction and optimization. *Journal of the ACM*, 44:201–236, 1997.

S.D. Bopardikar, F. Bullo, and J.P. Hespanha. A pursuit game with range-only measurements. In *Proceedings of the 47th Conference on Decision and Control*, pages 4233–4238, 2008. Cancun, Mexico.

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. 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*, 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*, 2011. In press.

C.A. Coello Coello. Evolutionary multi-objective optimization: a historical view of the field. *IEEE Computational Intelligence Magazine*, 1:28 – 36, 2006.

D. T. Cole. *A Cooperative Unmanned Aircraft System Architecture for Information-Theoretic Search and Track*. PhD thesis, University Of Sydney, 2009.

P. Dasgupta, P.P. Chakrabarti, and S.C. De Sarkar. *Multiobjective Heuristic Search: An Introduction to Intelligent Search Methods for Multicriteria Optimization*. Computational Intelligence. Vieweg, 1999.

T. Dean and M. Boddy. An analysis of time-dependent planning. In *Proceeding of the Second AAAI Conference on Artificial Intelligence*, pages 49–54, 1988.

R. Dechter. *Constraint Processing*. Morgan Kaufmann, 2003.

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*, 2008. Pasadena, USA.

M. Ehrgott and X. Gandibleux. A survey and annotated bibliography of multiobjective combinatorial optimization. *OR Spectrum*, 22:425–460, 2000. ISSN 0171-6468.

A. Farinelli, A. Rogers, and N.R. Jennings. Bounded approximate decentralised coordination using the max-sum algorithm. In *Proceedings of the IJCAI-09 Workshop on Distributed Constraint Reasoning*, pages 46–59, 2009. Pasadena, USA.

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*, pages 639–646, 2008. Estoril, Portugal.

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

T. Furukawa, F. Bourgault, B. Lavis, and H. F. Durrant-Whytekawa2006. Recursive bayesian search-and-tracking using coordinated uavs for lost targets. In *Proceedings of the 2006 IEEE International Conference on Robotics and Automation*, 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*, pages 2807–2812, 2007. Florida, 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(3):335 – 348, 1989.

C. Geyer. Active target search from uavs in urban environment. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 2366–2371, 2008. Pasadena, USA.

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

S. Harikumar and S. Kumar. Iterative deepening multiobjective A*. *Information Processing Letters*, 58:11–15, 1996.

P.E. Hart, N.J. Nilsson, and B.Raphael. Correction to "A formal basis for the heuristic determination of minimum cost paths". In *SIGART Newsletter*, volume 37, pages 28–29, 1972.

P.E. Hart, N.J. Nilsson, and B. Raphael. A formal basis for the heuristic determination of minimum cost paths. In *IEEE Transactions on Systems Science and Cybernetics*, volume 4, pages 100 –107, 1968.

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*, 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 Guidance, Navigation, and Control Conference and Exhibit*, 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.

G. Inalhan, D.M. Stipanovic, and C.J. Tomlin. Decentralized optimization, with application to multiple aircraft coordination. In *Proceedings of the Fourty First IEEE Conference on Decision and Control*, volume 1, pages 1147 – 1155, 2002. Las Vegas, USA.

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*, pages 2923–2928, 2004. Nassau, Bahamas.

J. Johannes. *Vector Optimisation.* Springer-Verlag, 2004.

V. Khare, X. Yao, and K. Deb. Performance scaling of multi-objective evolutionary algorithms. In *Evolutionary Multi-Criterion Optimization*, volume 2632 of *Lecture Notes in Computer Science*, pages 376–390. Springer, 2003.

R.E. Korf. Depth-first iterative-deepening: An optimal admissible tree search. *Artificial Intelligence*, 27:97–109, 1985.

M. Kovacina, D. Palmer, G. Yang, and R. Vaidyanathan. Multi-agent control algorithms for chemical cloud detection and mapping using unmanned air vehicles. In *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems*, pages 2782–2788, 2002. Lausanne, Switzerland.

A. Krause, J. Leskovec, C. Guestrin, J. VanBriesen, and C. Faloutsos. Efficient sensor placement optimization for securing large water distribution networks. *Journal of Water Resources Planning and Management*, 134(6):516–526, 2008.

V. Lesser, C. L. Ortiz, and M. Tambe. *Distributed Sensor Networks, A Multiagent Perspective.* Kluwer Academic Publishers, 2003.

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

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

R.T. Marler and J.S. Arora. Survey of multi-objective optimization methods for engineering. *Structural and Multidisciplinary Optimization*, 26:369–395, 2004.

P. J. Modi, W. M. Shen, M. Tambe, and M. Yokoo. Adopt: Asynchronous distributed constraint optimization with quality guarantees. *Artificial Intelligence*, 161(1-2):149–180, 2005.

A.H. Mong-Ying, A. Cowley, V. Kumar, and C.J. Taylor. Towards the deployment of mobile robot network with end-to-end performance guarantees. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA 06)*, pages 2085–2090, 2006. Orlando, USA.

A. Mouaddib, M. Boussard, and M. Bouzid. Towards a formal framework for multi-objective multi-agent planning. In *Proceedings of the Sixth International Conference on Autonomous Agents and Multiagent Systems*, pages 801–808, 2007.

V. Pareto. *Manual of political economy / by Vilfredo Pareto ;*. Macmillan, 1906. Translated by Ann S. Schwier and Alfred N. Page, 1972.

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

A. Rogers, A. Farinelli, R. Stranders, and N. R. Jennings. Bounded approximate decentralised coordination via the max-sum algorithm. *Artificial Intelligence*, 2011. In press.

E. Rollón. *Multi-Objective Optimization in Graphical Models.* PhD thesis, Universitat Politecnicá de Catalunya, 2008.

E. Rollón and J. Larrosa. Bucket elimination for multiobjective optimization problems. *Journal of Heuristics*, 12:307–328, 2006.

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.

S. B. Stewart and C. C. White. Multiobjective A*. *Journal of the ACM*, 38:775–814, 1991.

R. Stranders. *Decentralised Coordination of Information Gathering Agents.* PhD thesis, University of Southampton, 2010.

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*, pages 299–304, 2009. Pasadena, USA.

R. Stranders, F. M. Delle Fave, A. Rogers, and N.R. Jennings. A decentralised coordination algorithm for mobile sensors. In *Proceedings of the Twenty Fourth AAAI Conference on Artificial Intelligence*, pages 874–880, 2010.

P. B. Sujit and D. Ghose. Multiple uav search using agent based negotiation scheme. In *Proceeding of the 2005 American Control Conference*, pages 2995–3000, 2005. Portland, USA.

E. Talbi, S. Mostaghim, T. Okabe, H. Ishibuchi, G. Rudolph, and C. Coello. Parallel approaches for multiobjective 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.

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(9):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*, pages 1985–1990, 2008. Seattle, USA.

UAS Roadmap. Unmanned aircraft systems roadmap (2005 - 2030). Technical report, Office of the Secretary of the Defence, 2005.

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.

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*, pages 2948–2955, 2001. Seoul, Korea.

A. Waldock, D. Nicholson, and A. Rogers. Cooperative control using the max-sum algorithm. In *Second International Workshop on Agent Technology for Sensor Networks*, pages 65–70, 2008. Estoril, Portugal.

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*, pages 116–121, 2005. Portland, USA.

E. Zitzler, K. Deb, and L. Thiele. Comparison of multiobjective evolutionary algorithms: Empirical results. *Evolutionary Computation*, 8:173–195, 2000.