UNIVERSITY OF SOUTHAMPTON

A Combined Mechanism for UAV Explorative Path Planning, Task Allocation and Predictive Placement

by

Chris Baker

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

in the
Faculty of Physical Sciences and Engineering
Electronics and Computer Science

20th September 2016

UNIVERSITY OF SOUTHAMPTON

ABSTRACT

FACULTY OF PHYSICAL SCIENCES AND ENGINEERING
ELECTRONICS AND COMPUTER SCIENCE

Doctor of Philosophy

by Chris Baker

The use of Unmanned Aerial Vehicles (UAVs) is becoming ever more common by people or organisations who wish to get information about an area quickly and without a human presence. As a result, there has been a concerted effort to develop systems that allow the deployment of UAVs in disaster scenarios, in order to aid first responders with collecting imagery and other sensory data without putting human lives at risk. In particular, work has focused on developing autonomous systems to minimise the involvement of overstretched first responder personnel, and to ensure action can be taken by the UAVs quickly, co-operatively, and with close to optimal results. Key to this work, is the idea of enabling coordinated UAVs to explore a disaster space to discover incidents and then to allow more detailed examination, imagery, or sensing of these locations.

Consequently, in this thesis we examine the challenge of coordinating exploratory and task-responsive UAVs in the presence of prior (but uncertain) beliefs about incident locations, and the combination of their roles together. To do this, we first identify the key components of such a system as: path planning, task allocation, and using belief data for predictive UAV placement. Subsequently, we introduce our contributions in the form of a complete, decentralised system for a single explorative path planner to minimise the time to identify incidents, to allocate incidents to UAVs as tasks, and to place UAVs prior to new tasks being found.

Having demonstrated the efficacy of this solution in experimental scenarios, we extend the formulation of our explorative path-planning problem to multiple UAVs by constructing a coordinated, factored Monte-Carlo Tree Search algorithm for use in a discretised-space representation of a disaster area. Subsequently, we detail the performance of our new algorithm against uncoordinated alternatives using real data from the 2010 Haiti earthquake. We demonstrate the performance benefits of our method via the metric of people discovered in the simulation; showing improvements of up to 23% in cases with ten UAVs. This is the first application of this technique to very large action spaces of the type encountered in realistic disaster scenarios.

Finally, we modify our coordinated exploration algorithm to function in a continuous action space. This represents the first example of a continuous factored coordinated Monte-Carlo Tree Search algorithm. We evaluated this algorithm on the same Haiti dataset as the discretised version, but with a new sensor model simulating mobile phone signal detection to represent the types of sensors deployed by first responders. In addition to the benefits of a more realistic model of the environment, we found improvements in survivor localisation times of up to 20% over the discrete algorithm; demonstrating the value in our approach.

As such, the contributions presented in this thesis advance the state of the art in UAV coordination algorithms, and represent a progression towards the widespread deployment of autonomous platforms that can aid rescue workers in disaster situations and—ultimately—save lives.

Contents

List of Figures

List of Algorithms

Declaration of Authorship

I, Chris Baker, declare that the thesis entitled *A Combined Mechanism for UAV Exploratative Path Planning, Task Allocation and Predictive Placement* and the work presented in the thesis are both my own, and have been generated by me as the result of my own original research. I confirm that:

- this work was done wholly while in candidature for a research degree at this University;
- where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated;
- where I have consulted the published work of others, this is always clearly attributed;
- where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work;
- I have acknowledged all main sources of help;
- where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself;
- parts of this work have been published as: Baker et al. (2016a) and Baker et al. (2016b)

Signed: .....

Date: .....

Acknowledgements

Firstly, I'd like to thank my supervisors Nick Jennings and Sarvapali (Gopal) Ramchurn for their continued support (and unfailing patience) throughout my PhD and giving me the opportunity to work in the AIC group, and to Luke Teacy for repeatedly steering me in the right direction and being a priceless source of clear explanations of difficult concepts. Thanks also go to all my lab-mates, and in particular Oliver Parson, Sam Miller, and Moody Alam for passing on their wisdom and lessening the shock transition from undergraduate to postgraduate life.

I owe an immense debt of gratitude to all the various housemates, coursemates, and other friends I've spent my last few years with, including Ben Gray, Paul Gow, Sam Hendry, Joe Spencer, Alex Jantzen, Sami Kanza, and Sarah Jones. Special mention goes to the University of Southampton physicists that resisted the urge to stab me in the back for defecting to computer science: I appreciate your self control.

Thanks to Amr Hussein for being a great source of intellectual conversation even while drunk, until you fled to the other side of the world. Thank you as well to Hamid Khan for the frequent tea breaks and political discourse, until you also fled to the other side of the world. Many thanks to Bethany Growns for being funny, supportive, and a superb psychiatrist despite already being on the other side of the world. Thank you to Charlotte Parry for making my life about 28% more Welsh, and being a reliable and funny human being. Thank you to SJ Letcher for being far more understanding of my problems than I usually am, and also indirectly getting me free fish. A big thank you to Jo Carthy for a constant stream of inappropriate jokes, a steady and much needed supply of hugs, and the best impersonation of a gremlin I've ever come across. You've all made more of a difference to my life than you'll ever appreciate.

To my Mum, Dad, Brother, Sister and the rest of my family: thank you for the company, conversation, and help (both emotionally and sometimes financially) throughout my time at University; I hope I was worth it. As a small token of my gratitude you're under no obligation to actually read this thesis. To Alan and Lorraine Viles: thank you for helping me avoid homelessness for these last few months, and sometimes letting me use your kitchen.

Most importantly, thank you from the bottom of my heart to Georgie. Through a combination of affection, understanding, and some very creative insults you've kept me (mostly) sane through some particularly difficult times. Without your love and support I'd be bored, lonely, and probably even more grumpy. You're alright, kid.

Finally thanks to Old Mout cider, Costa Coffee, and my hamster Parsnip.

Nomenclature

a	Total joint action for all explorer UAVs: $a \subseteq A$
a_κ	Actions available to explorer UAV u_κ: $a_\kappa \in a$
A	The superset of all actions and joint actions available to explorative UAVs
C	Constant heuristic in K limit on continuous node expansion
c_{xy}	A single cell in a grid-world, located at (x, y)
C	Cost penalty for high-level explorer revisiting previous waypoints
ζ	Truncation factor for cost function C
ϱ	Heuristic weighting factor for C
ω_{prev}	Weighting factor in C
$\bar{d}_{xy}$	Expected danger value in c_{xy}: $\bar{d}_{xy} \in (0, 1)$
$\mathcal{D}$	Danger function as a 2D mapping of locations onto expected death rate: $\mathcal{D} : l \rightarrow \bar{d}_{xy}$
d_κ	Danger associated with signal s_κ, based on expected location: $d_\kappa \in (0, 1)$
d_{xy}	Danger present in c_{xy}: $d_{xy} \in (0, 1)$
Δ	Number of iterations of tree-search simulations
δ	The distance threshold below which an incident is considered discovered
∂	Denotes distance. For instance distance from i to a location l as: ∂_i^l
e	Current number of action expansions for a given node in MCTS
F	Global utility function for attending incidents
g	Location of all high-level explorers, with specific time indicated $g(t)$
g_κ	Location of high-level explorer u'_κ: $g_\kappa \in g$

G	Union of the locations of all high-level explorers: $G = \bigcup_{\kappa} g_{\kappa}$, with specific time indicated $G(t)$
μ	Mean of a given Gaussian component
σ	Variance of a given Gaussian component
i	A single incident: $i \in I$
I	Set of all incidents
j	The number of incidents: $j =\mid I \mid$
$\mathcal{J}$	Joint action tree creation function
K	Maximum number of actions per tree-search node in continuous MCTS
λ	Gaussian component of $\mathcal{M}$
$\mathbf{l}$	A location vector, in the form (x, y)
L	Location function returning (x, y) for (for instance) an incident i: $L(i) = (x_i, y_i)$
$\mathcal{L}$	Starting locations of UAVs
$\mathcal{M}$	Belief map representing probability of location of incidents
$\mathfrak{N}$	The set of all joint action trees
N	Function to return neighbouring nodes for max-sum coordination
n	An action tree such that $n \in \mathbf{N}$, corresponding to a variable node in a max-sum coordination factor graph
$n^{(\kappa)}$	A node in the tree-search corresponding to the actions in the variable n
ν	Maximum number of action expansions for a given node in MCTS
$\mathbf{N}_r$	The set representing the root nodes in $\mathbf{N}$ such that $\mathbf{N}_r \subset \mathbf{N}$
$\mathbf{N}$	The set representing the domain of the factor nodes to be coordinated in MCTS
$\mathcal{N}_2$	Two dimensional normal Gaussian distribution (with mean μ and variance σ)
η	Number of high-level explorative UAVs in the scenario: $\eta =\mid \mathcal{U}_h \mid$
O	A set of observations performed in the search space
o_{xy}	Binary variable denoting the visibility (or observed status) of c_{xy}: $o_{xy} \in \{0, 1\}$
ρ	The length of the trajectory of an explorative UAV: $\rho =\mid T \mid$
$\bar{p}_{xy}$	Expected number of people in c_{xy}

p_κ	People associated with signal s_κ, based on expected location
p_{xy}	Number of people in c_{xy}
ξ	Potential repulsion parameter
$P(i)$	Probability function: the probability of incident i being in an interval $dxdy$ can be expressed $P(i)dxdy$
P	Transition probabilities used in MDP formulation
q	Max-sum variable to function message
Q	Bellman equation utility value
r	Max-sum function to variable message
R	Global reward: at a specific time this is denoted $R(t)$
$\mathcal{S}$	A 2D map of the search area, for instance formed of a grid of cells
s	State of each cell, with time index denoted $s(t)$
s_{xy}	State of c_{xy}: $s_{xy} \in s$
$\tilde{s}$	State of the environment—cells and UAV locations—with time index denoted $\tilde{s}(t)$: $\tilde{s} \in S$
S	All possible states of the search environment
τ	A set of task allocations, one for each low-level UAV
$\mathfrak{T}$	The set of all possible task allocations: $\mathfrak{T} \ni \tau$
$\tau^u_{L(i)}$	Task allocation denoting uav u responding to task i at $L(i)$: $\tau^u_{L(i)} \in \tau$
θ	Temperature paremeter for simulated annealing
t	Time, expressed as an integer time-step
T_κ	Trajectory of high-level UAV u'_κ (with the subscript omitted in the special case of a single high-level UAV)
$\mathbf{T}$	The set of all trajectories T_κ
$\mathcal{T}$	Union of all trajectories
u'	A single high level explorative UAV: the apostrophe being omitted where distinction from task-attending UAVs is not required: $u' \in \mathcal{U}_h$
$\mathcal{U}$	The set of all UAVs in the disaster space: $\mathcal{U} = \mathcal{U}_h \cup \mathcal{U}_l$
$\mathcal{U}_h$	The set of all high-level explorative UAVs: $\mathcal{U}_h \subset \mathcal{U}$

$\mathcal{U}_l$	The set of all low-level task-attending UAVs: $\mathcal{U}_l \subset \mathcal{U}$
U	General notation for a Utility Function
γ	Truncation factor for U to prevent unbounded values
V	Potential field function used in prior-placement
v	UAV velocity
χ	The maximum x value, denoting the limit of the search area
ψ	The maximum y value, denoting the limit of the search area
(x, y)	Standard 2D Cartesian coordinates

Chapter 1

Introduction

The scene of a disaster, be it man-made or natural, is frequently one of damaged buildings, trapped victims, environmental hazards, and first responders attempting to negotiate the environment to help and rescue survivors. For these workers to perform their jobs expediently and as safely as possible, prompt information and sensing data of the area is essential for finding victims and safely navigating to their location. However, existing information from maps may be outdated or not reflect the situation on the ground after large scale geological disasters such as earthquakes or tsunamis. For example, during 2010 a magnitude 7.3 earthquake struck Haiti near the capital, Port-Au-Prince, which caused widespread destruction over thirty kilometres from the epicentre and resulted in the destruction or damage of over three hundred thousand homes; the death or injury of over five hundred and twenty thousand people; and around one point three million displaced persons needing temporary shelter (Government of the Republic of Haiti, 2014). More recently an earthquake in Japan in 2011 triggered a devastating tsunami along the country's East coast; killing around sixteen thousand people and causing between $14-35billion in damage (Kazama and Noda, 2012; NOAA, 2011). The 2015 7.8 magnitude earthquake in Nepal presented additional challenges to first responders and aid workers: the mountainous country has some communities and areas that are extremely isolated. As a result, it proved extremely difficult to identify the extent of damage to infrastructure and buildings, and to ascertain the needs of survivors (USAID, 2015). Even six months after the initial quake, areas of the country were still largely inaccessible, and over half a million people were without permanent shelter and facing acute food shortages (UNICEF, 2015).

With disasters of this scale—and with the resultant mass destruction of buildings and infrastructure—maintaining situational awareness of the scene of a disaster becomes important and challenging (Fowler, 2016). As such, there remains a vital need for ongoing sensing and imagery (for example mobile phone signal data, or thermal imagery) in real-time (Ehrlich et al., 2010; Fowler, 2016; OCHA, 2014; Harvard Humanitarian Initiative, 2011) as well as explorative efforts to actively seek out areas that require the attention

of emergency response workers. Currently, any thorough exploration of damaged areas is often carried out by personnel on the ground, introducing a safety risk to these workers and further draining the manpower that is vital to ensuring relief work is carried out properly. For example, during the World Trade Center attack in 2001, numerous members of the fire department were killed while searching for survivors as the towers collapsed (McKinsey & Company, 2004), and reports show that—even after disaster response has ceased—first responders can suffer from mental health difficulties following the event (Benedek et al., 2007; van der Velden et al., 2012).

Against this background, manned aircraft—namely planes and helicopters—have been used to increase the situational awareness of emergency services on the ground by providing prompt information where it is required over a larger area (for instance over a city, region, or entire country) or on a larger scale (e.g. high altitude imagery or extensive quantities of data). However, manned vehicles such as these are high-maintenance and not easy to deploy en-masse; especially as they require at least two operators: one to fly the aircraft and one to capture the imagery and communicate with those on the ground (Barnes et al., 2000; Cummings, 2013; Derbyshire Constabulary, 2013). Furthermore, such aircraft are very expensive to both procure and maintain and, although more manoeuvrable than fixed-wing planes, even helicopters are limited in their agility and ability to perform detailed exploration in cluttered environments. Thus, work is continually being undertaken to reduce the required manpower in disaster response, reduce the risk to the responders in the disaster area and improve information gathering about the area itself.

In the subsections that follow, we discuss possible solutions to these problems using unmanned vehicles, and the challenges faced in this research area (Section 1.1). We then go on to outline our contributions to the field: specifically in the form of a combination of three algorithms allowing for coordinated exploration, task allocation, and predictive placement of unmanned vehicles (Section 1.2). Finally we outline the structure of the rest of this thesis in Section 1.3.

1.1 Autonomous Vehicles in Disaster Response

Because of the problems and risks outlined above, disaster response work is increasingly being aided by unmanned robotic vehicles: either Unmanned Aerial Vehicles (UAVs) or Unmanned Ground Vehicles (UGVs). These provide a number of advantages and safety guarantees to those in the area: they are highly mobile and agile, are often cheap to purchase and deploy, and ensure operators are less exposed to the dangers of being in a disaster area (Adams et al., 2010; Measure, 2015; Tadokoro, 2009). For these reasons, unmanned vehicles are being actively encouraged in disaster response, seeing deployment following disasters like Hurricane Katrina and the 2010 Haiti earthquake, where

UAVs provided valuable post-disaster imagery (Adams and Friedland, 2012); and the 2011 Japan earthquake, where different UAVs provided thermal imaging and radiation sensing around the damaged nuclear plant at Fukushima (Murphy, 2012). Moreover, governments continue to encourage unmanned vehicle deployment through schemes such as the US DARPA programme offering financial incentives for UGV development in aiding first responders (US-Government, 2012), and the UK government's funding of UAV disaster response through the Engineering and Physical Sciences Research Council (EPSRC) (EPSRC, 2016; Orchid Project, 2014) and the ongoing Highly Innovative Technology Enablers in Aerospace (HITEA) programme (Innovate UK Technology Strategy Board, 2015).

The effectiveness of deploying UAVs can be seen in several case studies. For example, after the Haiti earthquake where a private company used a manually controlled drone to perform aerial imagery over an orphanage and determined that it was structurally sound (VT-Group, 2011). This work ensured emergency services were not required to perform a ground survey of the remote location and helped ensure resources were used more effectively where they were needed most. UAVs have assisted in assessing earthquake damage to buildings in Bologna (Italy) (Fernandez Galarreta et al., 2015), and Japan (following the 2011 Tohoku quake) (Yamazaki et al., 2015): this latter work focussing on using visual data collected by UAVs to construct three dimensional models of damaged structures to benefit the situational awareness and safety of rescue workers. The authors apply the same principles to data from the 2015 Gorkha earthquake in Nepal; a disaster where the aid response benefitted greatly from UAV imagery because of the remote and inhospitable mountainous locations (Brando et al., 2015; Yamazaki et al., 2016).

Although providing many advantages over manned flight, the operation of unmanned vehicles still requires significant user input. Typically, at least two people are required to operate a single vehicle: one to undertake full time piloting duties and another to examine the data feed from the onboard cameras or sensors (Cummings et al., 2009; Dixon et al., 2005; The IAFC Board of Directors, 2014). With this in mind, a report following the attacks of September 11th 2001 by Casper and Murphy (2003) recommends an ideal 1:1 robot to operator ratio, resulting in an increase in available free manpower. To help achieve (and indeed go beyond) this ratio, systems that are increasingly independent via *autonomy*—specifically, those capable of independent action and decision making—are required (Murphy, 2011). In particular, autonomy is an important concept in considering robotic systems for disaster relief, since it allows robots to be more self-reliant and less dependent on human intervention in decision making. The decisions the robots make could have a direct impact on saving lives. Relying on a human first responder to intervene in these decisions reduces some of the advantages of having a robot asset in the first instance, since it prevents that responder from being able to perform other important life-saving roles in the disaster area. Consequently, computer scientists have long been attempting to reduce the burden on operators and allow them to focus on

the vital ground work of disaster relief while aiding them in coordination and planning (Casper and Murphy, 2003; Murphy, 2012; Ramchurn et al., 2010). A further consideration, is that in a disaster relief scenario the first responders rely on working in teams to maximise their efficacy (in attending situations) and resources. For similar reasons, it is important that UAVs or UGVs do so as well; with teams of vehicles complementing each other to operate more effectively (Macarthur et al., 2011; Yang et al., 2004). For example, teams of UAVs with similar sensors can co-operate to reduce uncertainty in measurements; whereas teams with different sensors can compliment each-other to build up more cohesive information about a scenario (Ollero et al., 2006). Consequently, it is therefore essential that a system demonstrating autonomy must also allow for communication and co-operative team work to be carried out. In particular, this may be achieved by modelling UAVs and UGVs as autonomous *computational agents*.

Here, an *agent* is defined as an autonomous actor (such as a computer system, robot, or human) in some environment which must carry out autonomous actions to affect the environment (Wooldridge and Jennings, 1995), in order to meet its design objectives (Wooldridge, 2008). Moreover, in many circumstances this may require an agent to choose actions in the context of a multi-agent system with co-operating teams of agents, or self-interested agents in competition with each other. Multi-agent systems have been used to model UAVs in disaster response because of the clear mapping of UAVs to agents working in a collaborative team with some common goal (Hu et al., 2012; Sawhney et al., 2009; Scerri et al., 2005) (see Chapter 2 for more detail).

In particular, there are two well studied areas in the robotic-agent field, which are particularly important in disaster response, because they represent the key actions that must be taken to both gather and exploit information about the situation on the ground. These are the areas we focus on in our contributions:

Explorative Path Planning: This describes the generation of paths for a robot to follow through an environment. In our context, this refers to the exploration of the disaster area via informative paths, in order to discover incidents that need attending or people in need of assistance. Currently, the state of the art for UAV path planning algorithms focuses on three main areas:

- Target tracking for surveillance (Bernardini et al., 2014; Hu et al., 2014; Kolling and Kleiner, 2013): although these techniques are related to the exploration of a disaster space, they are designed to find a known number of targets that are in motion, rather than an unknown number of survivors distributed over an area.
- Trying to reach a set goal location (or state) (Chen et al., 2014; Desaraju and How, 2011; Durkota and Komenda, 2013; Guitton et al., 2009; Kothari et al., 2009; Levine et al., 2012).

- Exploring an area for the purposes of mapping the environment or minimising sensing uncertainty (Luotsinen et al., 2004; Sawhney et al., 2009; Singh et al., 2009; Sujit and Beard, 2009; Yang et al., 2013).

Instead, works such as Pineau and Gordon (2005) and Scerri et al. (2008) describe an explorative path planning algorithm which uses prior knowledge to *inform a path* to track a target or explore an environment. In contrast to the other approaches, this algorithm tackles path planning without a specific goal position in mind or a particular state at which to cease path generation (such as in the naïve exploration case, when the environment is fully revealed). Our work builds on these approaches by first showing the benefits of combining exploration with task allocation, detailed below. Secondly, we show an implementation for exploring a search space by coordinating the decisions made between multiple UAVs in planning their trajectories (co-operative exploration). Finally we extend the coordinated exploration algorithm to a domain with a continuous action space to sample and show that paths can be successfully coordinated in this way. Furthermore, making the model (and the algorithm) more realistic in this way delivers performance benefits over approaches where the problem is decomposed into discrete spaces.

Task Allocation: This describes the process of assigning tasks to a set of agents, with specific constraints. Specifically in our work, this encompasses the assignment of UAVs to discovered incidents, in order to return information– such as imagery or casualty data– to the first responders at the scene. Task allocation algorithms provide increasingly robust solutions to the problem of distributing tasks among agents according to specific constraints on their ability to complete tasks (such as time or battery restrictions), on their knowledge of the tasks' existence, and in order to minimise (maximise) some overall cost (utility) such as the time taken for task completion (Alighanbari et al., 2006; Delle Fave et al., 2012a; Ramchurn et al., 2010; Scerri et al., 2005). Typically these tasks are set by first responders, but in this thesis we will explore tasks being set by other autonomous agents—that is, other UAVs—to further reduce operator requirements.

The combination of explorative path planning and task allocation is important in providing a complete framework for deploying UAVs in a disaster response environment. As mentioned above, the two areas together represent the fundamental components of finding and dealing with incidents requiring imagery or other sensing. To this end, some work has been done which begins to address this problem. Specifically, Bellingham et al. (2001) detail a combination of path planning and task allocation by considering the problem of path planning as a component of reaching designated tasks in a situation where not all UAVs are capable of completing all tasks. In more detail, they focus on the problem of finding paths to enable UAVs to reach particular task locations, through an environment filled with obstacles that must be avoided. As such, although superficially

	Northrop Grumman RQ-4 Global Hawk	**PrecisionHawk Lancaster-5**	**DJI Spreading Wings S900**
Speed:	570km/h	79km/h (max)	57km/h (max)
Weight:	14,628kg (max)	3.55kg (max)	Approx. 6.8kg
Flight time:	34+ hours	45 minutes	18 minutes
Cost:	$222.7 million	Approx. $25,000[1]	Approx. $3,700[1]

FIGURE 1.1: Example of three UAVs with very different abilities and specifications, that have been used in disaster response operations (Da-Jiang Innovations Science and Technology Co., 2014; United States Government Accountability Office, 2013; PrecisionHawk, 2016; US Air Force, 2014).

a fusion of the two different areas, the path planning element merely exists to facilitate the completion of tasks rather than as separate process for exploration.

We believe that existing work such as the example above has yet to address a fundamental property of unifying exploration and task allocation: namely that the two areas are suited to different platforms. Explorative path planning requires quick coverage of large areas and is best suited to faster UAVs with an extended operating range; responding to task allocation requires extended placement over a certain location to relay images to first responders, best suited to rotary UAVs which are capable of hovering at a location—possibly at low altitude—to obtain more detailed imagery. Fittingly, there are very distinct classes of UAV available: from large scale, high payload, long range and long endurance fixed wing drones of the type frequently used in military operations, to smaller, cheaper rotor-based UAVs with more dynamic flight control and loitering ability. This distinction is clearly shown in Figure 1.1, which compares a US military fixed-wing UAV, a smaller commercially available fixed wing UAV, and a commercially available rotor vehicle. All of these disparate vehicles have been used in disaster relief work (Adams and Friedland, 2012; Da-Jiang Innovations Science and Technology Co., 2014; Murphy et al., 2015); indeed, increased use of UAVs in disaster response will necessarily result in cases where heterogeneous groups of vehicles will need to work together for common goals. As a result, we seek to create the framework of a set of algorithms that is required to deploy systems of heterogeneous UAV platforms; taking the abilities of different types of UAV and using them as effectively as possible. Specifically, we seek to develop algorithms to control and coordinate UAVs of differing types to effectively perform exploration of a space and execution of discrete tasks.

[1]The price is dependent on chosen options and attachments.

As well as using UAVs to gather information about a disaster area, the pervasiveness of mobile communication equipment has allowed for crowd-sourcing to become useful in delivering knowledge of incidents or emergencies to first responders, subject to the reliability of the sources, uncertainty in locations, and availability of communication networks. Two examples of such work are the Ushahidi platform, which uses submitted emails and SMS messages to construct a map of incident locations and was implemented in the Haiti earthquake of 2010 (Morrow et al., 2011); and the Crisis Mappers group, a collaborative network of volunteers who use mobile and web applications to collate and map information received during a disaster (Crisis Mappers, 2013). There is a growing body of work focussing on the accuracy (or otherwise) of information obtained via crowd-sourcing (Mathibela et al., 2012; Venanzi, 2014; Venanzi et al., 2015); for our purposes, we are content to assume that the information is accurate, and gives a satisfactory indication of the situation on the ground.[2] Henceforth, data distributed over a disaster area reflecting the likelihood of some ground-truth (for instance danger, incidents, people etc.) located in the space will be referred to as *belief maps* (in line with the terminology used in Bry and Roy (2011); Gan et al. (2012); Waharte et al. (2009) and others). These can be envisaged as contour maps over a search space; we assume it has the characteristics of mapping spatial locations onto some function that represents numerical data, which may or may not be normalised (some examples are shown in Figure 1.2).

Belief maps are one example of the types of data used to construct paths through an environment. In general, for path-planning to be non-trivial one expects the paths to be planned based on some external information from sensors, or beliefs about the environment. Nonetheless, even with prior information about an environment there are specific challenges of planning trajectories: notably the large number of actions available to UAVs (the so-called *action space*) that must be considered during the planning of paths (Huang and Gupta, 2008). This can quickly lead to solutions that are computationally intractable to calculate, if the number of possible states or actions to be planned over are excessive.

In light of these challenges, in our contributions we consider specific synergies between belief maps showing perceived danger to human life (namely a numerical representation of the likely death rate of persons in a given area in unit time), and probability distributions of the population of the disaster environment. This allows us to focus the exploration and path planning of UAVs on the largest expected number of *people*, in the highest expected amount of *danger*—for instance the expected rate of fatalities due to structural damage to buildings or from the spread of radiation—as the highest priority for visits by UAVs. The motivation here, is that once emergency responders are aware of the location of casualties they can ensure faster rescue efforts, and greatly increase their chance of survival (Macintyre et al., 2011). We illustrate this combination of beliefs in Figure 1.3.

[2]We discuss the implications of this briefly in Section 3.1.

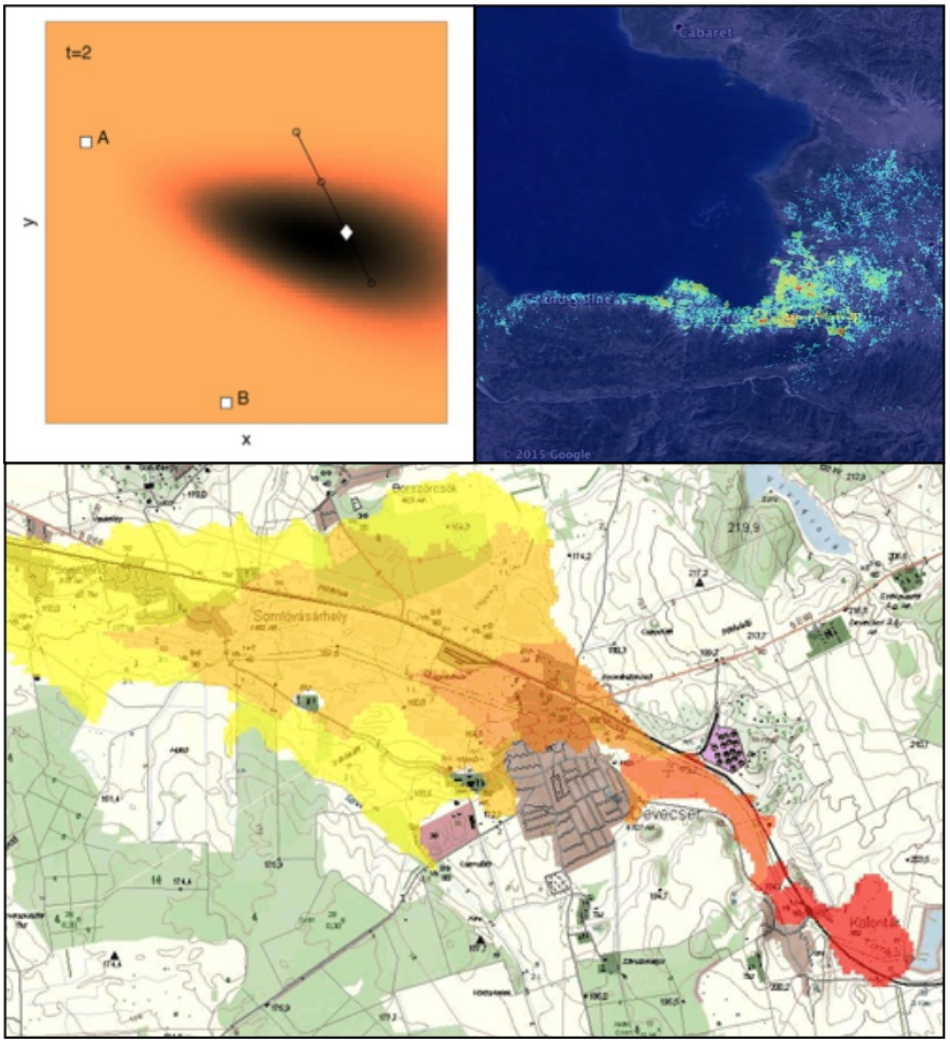


FIGURE 1.2: Examples of belief maps, clockwise from top left: a simple belief map used in a target tracking problem, darker areas indicating high belief and diamond showing true position (Makarenko and Durrant-Whyte, 2006); map of building damage over the capital of Haiti, Port-au-Prince generated from crowd reports following the 2010 earthquake there (Ush, 2015); projected map of the extent of toxic mud spillage in 2010 near the town of Ajka, Hungary (Police of Hungary, 2010).

Furthermore, we examine and note contributions in the field of explorative path planning that enable us to develop algorithms concerned with multiple high-level explorative agents capable of coordinating their actions in explorative path planning, by utilising this belief map data. We note that some work exists on coordinating UAVs with no *a priori* belief information (Yang et al., 2004) or synchronising arrival times between multiple travelling UAVs (Mishra et al., 2014). Nonetheless there remain several challenges to coordinating the exploration of multiple UAVs where belief data is present, in order to minimise casualties in a disaster scenario; for instance dealing with the large communications overheads of sharing information between UAVs (Waharte and Trigoni, 2010) and the complexity of the large action-space available with a team of vehicles (Ceccarelli et al., 2009). To address these challenges, we consider a *Monte-Carlo Tree Search* algorithm as a suitable candidate for the exploration component of our solution (see Section 2.3.4.1)

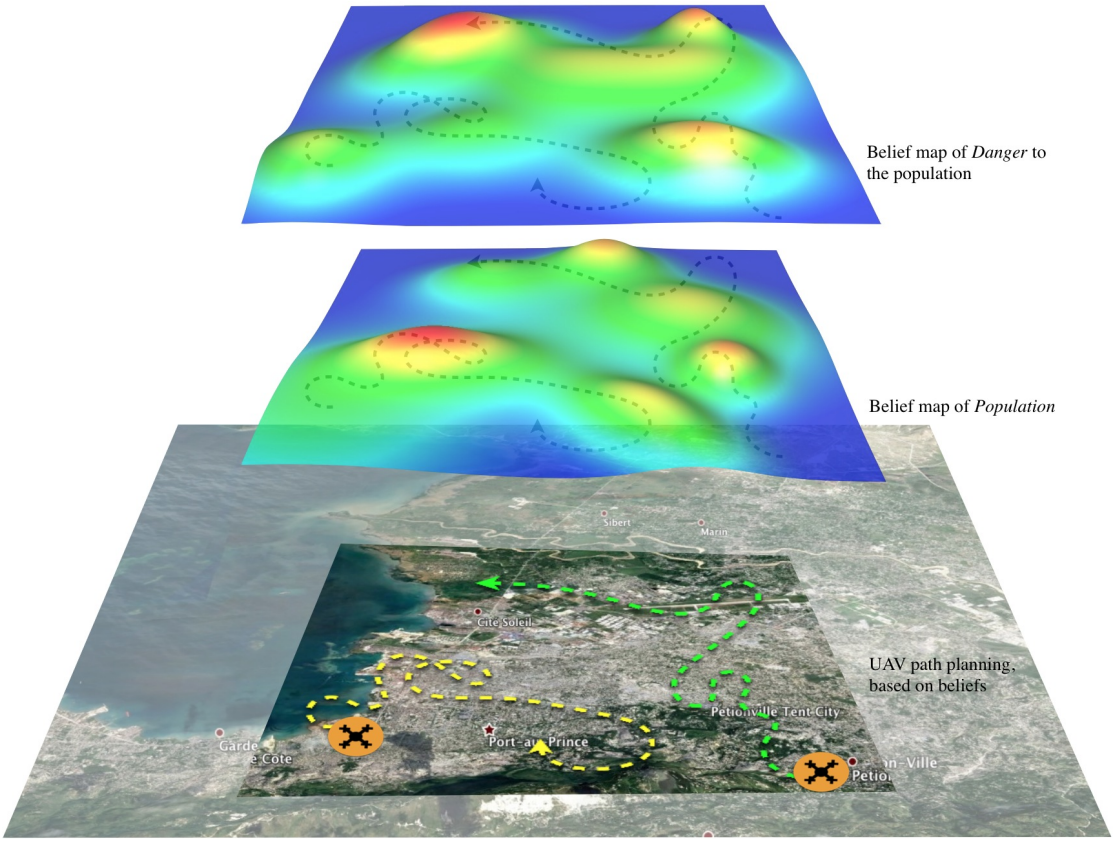


FIGURE 1.3: Example of belief map construction and UAV path planning from information on expected population distribution and the danger to those people.

given its established history in path planning (Cameron et al., 2010; Otte, 2011; Powley et al., 2012; Zaera et al., 2001) and the ability to easily factor in coordination at the stage where the algorithm simulates—in a step-wise manner—a predicted path through the disaster space.

Additionally, to further exploit the potential for UAVs to perform operations without human intervention, we considered the usefulness of belief maps for the predictive placement of UAVs in a disaster space to anticipate the location of discovered incidents to be attended, to reduce future travel time to incidents. Such work has not been attempted before in this specific context, but lessons can be drawn on from numerous fields dealing with geometric problems of structure (Nilges et al., 1988; Wille and Vennik, 1985a,b) and on sensor placement (Baras and Tan, 2004; Lin and Chiu, 2005; Raissi-Dehkordi et al., 2004; Salapaka et al., 2003). We do this, while bearing in mind the desired outcomes of our particular scenario: ensuring UAVs are placed to reduce travel time to the likely location of incidents.

In summary, this works seeks to develop and investigate UAV coordination algorithms that fulfil the following criteria:

1. The algorithms should ensure the abilities of UAVs are fully exploited. Specifically the fact that some vehicles are more suitable for loitering and long–time imaging, and that others are in general quicker and better at covering ground must be considered.

2. Within this context, discovered incidents should be allocated as tasks to a team of task-performing UAVs that must attend them as quickly as possible. Based on the belief maps, pre-placement of the imaging UAVs should take place to reduce response times. Exploratory UAVs should be informed by available belief maps, and should use this data to construct exploratory paths in order to best locate these incidents.

3. We require that we develop further exploration algorithms to account for *multiple* exploratory (sensor equipped) UAVs; requiring coordinated path planning across the belief data available.

Additionally, the work should fulfil the following research requirements that function as constraints on the implementation of the algorithms outlined above:

R1: **Decentralised Autonomy** The agents should act autonomously; that is each agent should be responsible for its own actions. This also encompasses the requirement that the system be decentralised, in order to remove a central point of failure for the system. This is particularly important for disaster situations where unreliable infrastructure and scarce supplies means lost resources cannot necessarily be replaced.

R2: **Coordination** Agents must act co-operatively to coordinate their actions as a team in order to maximise their performance: specifically in exploring and attending incidents as quickly as possible. Therefore, algorithms must focus on achieving a global objective rather than the myopic goals for each agent.

R3: **Scalability** The work should be scalable, and not face intractable overheads for communication and calculation with increased team size (up to tens or hundreds of agents, in order that the algorithms remain applicable in future disasters when UAV deployment is common).

1.2 Research Contributions

In the context of the advantages and goals described above, the contributions of the work described in this thesis are as follows:

1. We present a set of algorithms for both the exploration of a space for incident sensing and subsequent allocation of agents to discovered incidents for the purposes of further sensing, by considering heterogeneous UAVs in the roles of a path-following explorer and an incident responder. This holistic approach meets criteria 1 and 2 above. Criterion 2 has been met by developing a new prior placement algorithm to move UAVs towards areas where there is believed to be incidents, before assignment takes place. This reduces the time between incident discovery and imagery being returned to ground crews.

2. Following our work on the creation of a joint path-planning and task-allocation paradigm, we present the first application of a *factored Monte Carlo Tree Search* planner to a UAV exploration problem: also representing the first time this type of planner has been used in a very large state space ($\sim 10^{20}$ states). We use this for co-operative path planning by coordinating the planning of multiple exploratory UAVs in a discretised action and state space environment. Specifically this merges the system Criterion 3 calling for multiple exploratory UAVs, with Requirements R2 and R1 necessitating coordination and decentralised autonomy. We published these results in Baker et al. (2016a).

3. Finally, we remove the assumption of a discretised exploration space, and present the first *continuous factored path planning algorithm*, which we use for coordination. As well as meeting Criterion 2, Requirement R2, and Requirement R1; a continuous approach improves performance over the discrete case, as well as better meeting Criterion 2 since it no longer requires discretisation of the belief data available. We published these results in Baker et al. (2016b).

All work has been carried out in simulations, primarily because of regulatory difficulties fielding autonomous UAVs outside of governmental work. These contributions represent a highly beneficial approach to deploying UAVs in disaster response scenarios by reducing the burden on first responders, and increasing awareness of the environment by providing important sensory information about a disaster area.

1.3 Thesis Outline

The structure of the remainder of the thesis is as follows:

- In Chapter 2, we outline related work and literature, with a focus on the research areas and goals addressed by Section 1.1.

- Chapter 3 explores the contributions outlined in Section 1.2 in detail for a combined predictive placement, path planning and task allocation mechanism. In particular, we detail the algorithms developed; together with an empirical evaluation of the

performance of these mechanisms against various benchmarks. We show that the set of algorithms we have used perform at a minimum 25% better than other combinations of exploration, task allocation, and prior placement methods.

- Chapter 4 expands on the exploration problem introduced in the previous chapter to describe a mechanism for coordinated UAV exploration in order to coordinate multiple explorative agents in the search space.
- Chapter 5 introduces a new environment model to account for a more realistic sensor scenario, and removes the assumption of a discretised exploration space with a new continuous-space coordinated exploration algorithm. This introduces performance benefits over the discrete case of up to 20%, as well as reducing the burden on first responders of having to process data into a discrete cellular form before planning occurs.
- Finally Chapter 6 summarises the thesis and highlights areas of future work.

Chapter 2

Literature Review

In this chapter, we review the literature associated with the three main processes required in the execution of the criteria detailed in Section 1.1; namely path planning, task allocation and prior placement of UAVs. We highlight previous work undertaken in each area and applications relevant to our research goals, as well as any limitations or shortcomings that might make an approach unsuitable, or otherwise fail to meet the research goals (listed in Section 1.2).

2.1 Task Allocation

Task allocation is the process of distributing a series of tasks among multiple agents to maximise some overall utility function, which defines the relative value of one possible solution over another. The nature of a "task" is specific to each real-world situation, as is the utility being maximised. In the context of disaster relief, tasks typically involve returning imagery or sensory data from a specific location at the request of emergency workers; with the utility being a function of quantitative values relevant to the performance of the UAVs. Battery life, distance to target, and priority of the task are typical components of such a function; functioning as constraints: that is, limits on the performance of the UAVs. In what follows, we will consider each method against its ability to reduce the total time for task attendance, while also considering the other requirements from Section 1.1: namely decentralisation and scalability. We note briefly, that according to the nomenclature introduced in Gerkey and Mataric (2004), we will specifically consider task allocation problems that are *single task* (one task per agent) *single robot* (one agent per task) *instantaneous assignment* (assignment is carried out immediately and we do not explicitly consider reallocation): this is outlined in depth in Section 3.1 in the following chapter.

While work has been done towards task allocation algorithms using centralised methods such as integer programming (Richard, 1982) and simulated annealing (Attiya and

Hamam, 2006), we will here focus on methods that are explicitly tailored to decentralised approaches in keeping with Requirement R1. These represent the current state of the art in this field: auction based algorithms, negotiation algorithms, and algorithms that solve a formulation known as *decentralised optimisation problems*.

2.1.1 Negotiation-Based Task Allocation

Negotiation-based methods use a message passing framework between agents to allow them to negotiate to reach an agreed plan, and as such, fit into the broad category of planning algorithms. Planning algorithms can function as task allocation methods with correct treatment: namely assuming a specific plan corresponds to a specific assignment of agents to tasks. In this context, negotiation-based plan selection describes a method of planning based on negotiation between agents via a propose-and-accept message passing framework, introduced in a paper by Sujit et al. (2006), and expanded on in works such as Settembre et al. (2008), Binetti et al. (2013), Goldingay and van Mourik (2013), Liekna et al. (2012) and Kong et al. (2014). The goal of these systems is to provide a decentralised method of choosing a plan to maximise team utility, through allowing agents to converge on plans by sharing only the information necessary to justify their decisions. The environment is regarded as being in one of a series of different *situations*, each of which maps to a corresponding specific, desirable plan from a list of available plans. Since the exact situation of the environment is only known within a degree of certainty, the planner must balance selecting more effective *specific* plans for specific situations, or more general plans that carry a lower penalty if the environment does not exactly match the sensory data.

A broad overview of negotiation allocation is as follows. Robots initially select a plan which, according to their local information, maximises expected reward for the team. This is achieved using a simple decision theoretic calculation by picking a plan P from the (globally known) set of all plans $\mathcal{P}$. In order to confirm or refute that this is the best course of action given the knowledge of other agents, a process of negotiation via message passing occurs that allows the proposal of possible alternative plans if there exists information to support such a course of action. The message passing protocol proceeds as follows. When a robot receives a plan proposal from a neighbour, it checks to see if its own observations support the plan. If they do, it increases an agreement variable by one, indicating that another agent agrees with the proposal, and passes the message randomly to another agent. If the robot has observations or data that contradict the proposed plan, it changes the status of the message to *challenge* and returns it to the sender along with the observations which support its disagreement. On the other hand, if a challenge is received the robot integrates the observations included in the message with its own, and reconsiders its choice of plan. Two outcomes are then possible. Either the observations do not change its beliefs sufficiently to warrant a change of plan, in

which case the message is changed back to a proposal and sent on with the supporting evidence. Otherwise, it no longer considers its plan as the best course of action. When the number of agents agreeing to a plan reaches a predetermined value referred to as the TTL (time-to-live), the plan is carried out.

Our scenario can benefit from this negotiation approach; with similarities that can be seen if one considers the distribution of tasks to be attended as a situation, with different assignments of UAVs to different tasks as specific plans. Although we do not consider uncertainty in task positions, future work may consider situations where agents could be in disagreement about a plan due to incomplete local knowledge. Such situations would benefit from a negotiation phase allowing UAVs to share only those observations that are relevant to each agent in turn (i.e. the existence of other tasks).

Nevertheless, there are four main disadvantages from a negotiation-based method:

1. The requirement detailed in Section 1.1 calling for the system to be scalable is not guaranteed with this approach. Specifically, this work does not exclude the prospect of subsets of the agents agreeing on conflicting plans independently of each other. Although Settembre et al. (2008) claim such situations have been examined in Scerri et al. (2004), this latter work only addresses the potential conflicts arising from overlapping subsets of agents being allocated simultaneously to the same task: a situation not applicable to a case where agents are considered on an individual basis.

2. The system is not robust to changes in the structure or availability of agents while planning is underway (since the success of the algorithm depends on a predetermined TTL); potentially resulting in impossible or obsolete plans once implementation of the allocation has been achieved. In our context, as more tasks are discovered reassignment may be required to maintain quality of solutions.

3. The method may incur high message overheads to reflect the large amount of information to be conveyed (particularly the relevant supporting data required to justify a challenge), as well as a potentially quadratic relation between messages passed and the value of TTL (Settembre et al., 2008). Both of these undermine the method's usefulness when considering requirement R3 (scalability).

4. In a situation such as ours where we wish to reduce the overall time for agents to attend tasks, this framework lacks the explicit ability to compare quantitative information since it is presented in a high-level fashion with abstract reference to "evidence" and "observations", making mapping to real-world problems unclear. A more quantitative algorithm, comparing numerical values, would allow easy inter-agent comparison of times to attend tasks.

For these reasons, we consider negotiation unsuitable for our task allocation algorithm.

2.1.2 Auction-Based Task Allocation

Similarly to negotiation-based approaches, auction-based task allocation uses a proposal and acceptance framework to allow agents to allocate tasks, but with quantitative measures of allocation value considered as a vehicle for bidding to occur. Early work in this area includes the Contract Net Protocol outlined by Smith (1980) for distributing problem solving tasks among nodes on a *factor graph* (each node representing an agent) via a high-level bidding procedure. Factor graphs (as the name suggests) are graphs representing the factorisation of the objective function of a problem; typically via a bipartite *function* and *variable* node model (Kschischang et al., 2001).

A similar approach to Contract Net has been used more recently in resource allocation problems (Grosu and Das, 2004). These works provide a very broad approach to the problem, and are typically unconcerned with constraints such as unreliable communications that might result in incomplete node graphs that do not connect all agents to all other agents. The procedure is simple, and is constructed on the representation of agents in communication with each other as nodes in an undirected factor graph. A task is submitted by a node (representing an agent) for neighbouring nodes in the graph to bid on its completion. This is only permitted if the agent holding the task predicts a minimum of 10% (a heuristic chosen by the author) utility gain for the group by doing so (and not undertaking the task itself). The task is then awarded to the highest bidder; that is, the agent with the highest predicted individual utility for completing the task. A variation of this is that tasks are distributed by some form of centralised base, and coordination takes place between agents (assuming, that they are in communication) (Johnson et al., 2016).

The main drawback of this method is the exclusion of non-local data in the bidding process. That is, agents need not consider the actions of vehicles not directly connected to a given bid. In contrast to other allocation algorithms, this behaviour of auction allocation is undesirable as it gives rise to situations where tasks are not allocated optimally as bidding agents are unaware of their impact to other areas of the factor graph (i.e. other agents). In this sense, the algorithm exhibits some characteristics of greediness, since tasks may be allocated in the best interests of single agents or small groups of agents, rather than the entire group as a whole. To address this, Lemaire et al. (2004) introduced a method of sharing information about the agents' respective workloads, via a value called the *equity coefficient*. This value is negative if an agent has few tasks in its plan compared to other neighbouring agents, and positive if it has much more work. This can then be integrated into the utility value of an agent for task completion to ensure it is not overloaded by tasks. A further drawback in the original method tackled by this paper is the timing of the creation of auctions: concurrent auctions can result in sub-optimal allocation or agents winning the bid on two mutually exclusive tasks simultaneously. Lemaire et al. deal with this by using a token-passing method. Here, only

the agent holding a token can initiate auctions for tasks that need completing, and may receive requests from other agents to have the token passed on. This is achieved via random allocation weighted by the requesting agents' equity coefficient values, ensuring that overloaded agents are more likely to receive the token and be able to auction one of its tasks to other agents.

In spite of the improvements outlined and in work since (such as Feo Flushing et al. (2016); Liekna et al. (2012); Luo (2014)), auction-based task allocation still suffers some fundamental drawbacks. For example: since in the scenarios we develop in Chapters 3 and 4 beliefs about the search environment are updated based on UAV observations, we also consider the problem that by the time auctions have concluded, the data might be obsolete: resulting in plans that no longer reflect the best course of action for the situation in the environment. Additionally, work by Amador et al. (2014) concedes that even in their particular scenario—in which their auction-based allocation algorithm outperforms several standard benchmarks—the algorithm suffers notable shortcomings in replanning on discovery of new tasks, or when the number of tasks was large.

Most importantly, communication overheads remain a problem, and although the work on equity coefficients improves algorithm performance, it adds to the volume of messages being passed, reducing the ability of the system to meet the method requirement of scalability R3. Indeed, a review by Gerkey and Mataric (2004) concludes that for systems with up to around 200 agents, a centralised broadcasting approach for auctioning would incur *lower* communication costs than a decentralised solution (including considered latency). However, this clearly is not a feasible approach if we wish to meet method Requirement R1 requiring decentralisation.

In order to overcome these problems and ensure performance is not diminished in a completely decentralised system, we now consider task allocation as an optimisation problem to be solved among the team of agents.

2.1.3 DCOP-Based Task Allocation

It is possible to model the task allocation problem as a *decentralised* (or distributed) *constraint optimisation problem*, or DCOP. Such problems are characterised by the presence of a group of agents controlling the value of decision variables in a system, with a view to jointly optimising the global reward in the presence of constraints on the values of the variables (Farinelli et al., 2010). Specifically, such problems can be represented by graphs of variable and function nodes, with variable nodes representing the choices available to each agent and the function nodes returning numerical values to reflect the utility or cost to the system. If we consider an agent's choice as being its allocation to a task, and the functions as representing the global utility to be maximised, we have a direct mapping to the task allocation problem we seek to solve. Given this observation,

DCOP solutions have been widely studied with examples such as ADOPT (Modi et al., 2003) and DPOP (Petcu and Faltings, 2007) receiving particular attention. However, as argued by Farinelli et al. (2010), DCOP solutions suffer two main problems depending on whether they are optimal or approximate. Optimal solutions often carry high complexity or message overheads (e.g. ADOPT message overheads and DPOP memory complexity both increase exponentially), while approximate solutions (such as MGM (Maheswaran et al., 2004) and its extensions) do not carry performance guarantees. An exception to this argument is the *max-sum* algorithm (Farinelli et al., 2008), which carries both performance guarantees (in particular forms) and a design to mitigate large message costs.

Unlike negotiation and auction based allocation, the max-sum task allocation algorithm is designed explicitly to reduce message overheads between agents, while providing a robust and scalable framework for task utilities. This makes it particularly relevant to our work. Max-sum can be broadly described as a decentralised message passing framework that allows nodes to build local functions depending only on variables they control, to maximise some global utility (Farinelli et al., 2013). Specifically, the algorithm can be deployed over an undirected factor graph $\mathcal{FG} = \{\mathcal{N},\mathcal{E}\}$, where $\mathcal{N} = \mathcal{X} \cup \mathcal{F}$ is the set of all variable nodes $\mathcal{X}$ and function nodes $\mathcal{F}$; and $\mathcal{E}$ is a set of edges joining variables to functions and vice versa (Macarthur, 2011). An example is shown in Figure 2.1 of three variable nodes (representing agents) and three function nodes (representing tasks). This structure means that in some instances the physical arrangement of a network of communicating agents does not map directly to the max-sum factor graph, since additional nodes might need to be created. The messages passed between adjacent nodes can be thought of as follows. From function to variable, messages describe the best outcome for the other neighbours connected to that function, if the variable adopts any of its discrete options. Variable to function messages return the sum of all other received messages across the various different variable states. An example of a system where this algorithm can operate effectively is in the classic graph-colouring problem, where the variables represent the colour changing nodes and there are functions situated between them in the factor graph describing the utility value of neighbouring nodes depending on their colour. Work by Farinelli et al. (2008) shows max-sum applied to this scenario performing better than standard benchmarks,[1] while carrying small communication overheads since messages include only pairs of values.

In a task allocation context, the nature of a variable can be simplified to reflect whether or not a task is attended by that node (i.e. by the agent in possession of the node), leading to a variation known as fast max-sum (Ramchurn et al., 2010) or task allocation max-sum. Variable nodes are mapped to agents capable of attending tasks, with tasks represented by function nodes. The messages are simplified since messages from functions to variables need not consider information on every possible assignment of that variable

[1] Namely the Best Response, Distributed Stochastic Algorithm, and DPOP algorithms.

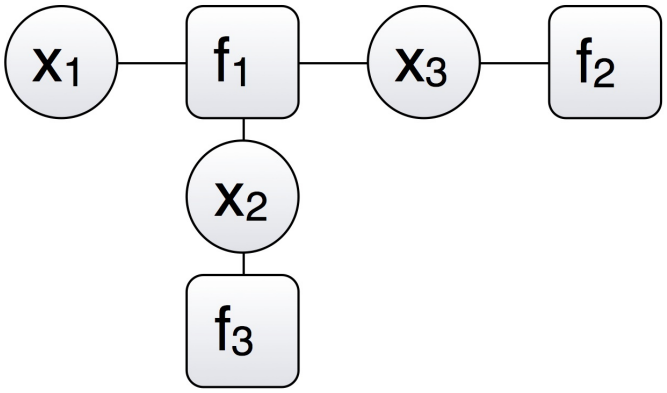


FIGURE 2.1: Example of a max-sum factor graph, with variable nodes (x_i) representing agents, and function nodes (f_j) representing tasks.

(i.e. every possible task it could attend), but rather only the maximum possible utility if it does attend to the function's task, or if it does not. The global utility function can be generally defined as the sum of each function f_j over each of the sets of variables connected to that function $X_j \subseteq X$:

$$F(X) = \sum_j f_j(X_j) \tag{2.1}$$

We now describe the form of the exchanged messages in detail by showing the composition of the messages from variable to function and from function to variable (in which i, j and k serve as indices for the various nodes):

- From variable $x_i \in X$ to function f_j, where $f_j : X_j \to \mathbb{R}$ represents the utility of assigning the set of connected agents X_j to a given task v_j (or conversely, to any other task except v_j, denoted as $v_{\neg j}$) :

$$q_{i\to j}\left(x_i = v_j\right) = q \quad \text{and} \quad q_{i\to j}\left(x_i = v_{\neg j}\right) = q'$$

  The composition of these messages is as follows, where $r_{_{k\to i}}$ represents the most recent message received by variable x_i from function $f_k \in \mathcal{M}_i$, and $\mathcal{M}_i$ is the set of functions connected to x_i:

$$q_{i\to j} = \alpha_{ij} + \sum_{k\in\mathcal{M}_i\setminus j} r_{k\to i}\left(v_{\neg k}\right)$$

$$q'_{i\to j} = \alpha_{ij} + \max_{v_b\neq v_j}\left[r_{b\to i}\left(v_b\right) + \sum_{k\in\mathcal{M}_i\setminus b,j} r_{k\to i}\left(v_{\neg k}\right)\right] \tag{2.2}$$

  where v_b represents a task to which x_i can be allocated apart from v_j. The α_{ij} values are normalisation constants that enable convergence in cyclical graphs (see below). The use of only two values in the message, q and q', reflects that in one

instance the variable is assigned to the function's task, and in all other instances the function does not benefit from the variable's assignment, regardless of what other task it undertakes.

- From function to variable:

$$r_{j\to i}(x_i) = \max_{\mathbf{x_j}\setminus i}\left[f_i(\mathbf{x}_j) + \sum_{k\in\mathcal{N}_j\setminus i} q_{k\to j}(x_k)\right] \tag{2.3}$$

where $r_{j\to i}(x_i)$ for $x_i = v_j$ and any other $x_i = v_{\neg j}$, and $\mathcal{N}_j$ is the set of variables connected to f_j. As above, this restricts the outcomes considered between cases where x_i undertakes v_j and where it does not.

Ramchurn et al. (2010) argue that this message passing process is equivalent to pruning the max-sum search space by removing searches of assignments that do not improve the utility of the function node in any way. Once variable nodes have received messages from all connected function nodes—namely all members of $\mathcal{M}_i$—they can compute their local portion of the objective utility function:

$$z_i(x_i) = \left(r_{j\to i}(v_j) + \sum_{k\in\mathcal{N}_j\setminus i} r_{k\to i}(v_{\neg k})\right) \tag{2.4}$$

The final assignment of x_i is the task which then maximises $\arg\max_{x_j} z_i(x_i)$. As with traditional max-sum, this can be run continually to allow the constant re-computation of solutions; making it robust to changes in the factor graph, allowing reassignment of UAVs as new tasks are discovered. Note that the utility function relating variables to their assignments is not explicitly shown here as its particular form does not affect the operation of the algorithm. Even this implementation has seen further improvement in scenarios with complex graphs with large numbers of agents by rendering the problem in terms of binary factors (Penya-Alba and Vinyals, 2012; Pujol-Gonzalez et al., 2015) (although in our scenarios the graph complexity was simple enough that additional pre-processing of the scenario into the more complex binary form was deemed unnecessary).

The task allocation version of max-sum was used explicitly to coordinate UAVs in Delle Fave et al. (2012b); first in simulation and then in a real-world scenario with two hexacopters. The utility function was designed to account for a number of constraints in the system, such as task distance, task importance, and remaining UAV battery life. In simulation, it was shown that the max-sum algorithm outperformed greedy assignment methods and those which had incomplete utility functions, demonstrating the applicability of this method to situations with multiple constraints. It was also found that since messages are composed entirely of pairs of values, and that messages are only exchanged locally, the algorithm scales very well (enabling real-time performance) with the addition of tens or even hundreds of agents in simulations, satisfying requirement R3.

In the context of our goal of assigning tasks specifically to UAVs, it is instructive to note the successful performance of the max-sum method in a real-world scenario. The paper details an experiment where tasks were submitted from PDA devices with a user interface to allow placement of tasks on a map of an area, while specifying the task's urgency and priority. The experiment was carried out on two agents in situations with homogenous tasks, sequential arrival of tasks, and heterogeneous tasks. Task assignment proceeded as expected and the authors note that future work will focus on scaling the complexity of tests, with more UAVs dealing with an even larger proportion of tasks (as one might expect in a real disaster).

From this work, we find that task assignment via a max-sum method matches our relevant requirements well, and also carries some well-documented performance guarantees. The max-sum algorithm is guaranteed to converge to an optimal assignment solution in acyclic factor graphs (Farinelli et al., 2010). With cycles, the main disadvantage is that convergence is no longer guaranteed due to the possibility of a continual increase in message values passed around loops in the graph. Possible methods for avoiding the continual increase of exchanged message values are to either introduce a normalising constant (as shown above, with the constant between x_i and v_j denoted as α_{ij}) to the messages, or to prune the factor graph into a tree. The latter can provide solution guarantees on the affect to the overall utility due to the pruning. Nevertheless, cyclical graphs with normalisation have empirically been shown to produce very close to optimal results and converge in all studied cases; and carry a smaller overhead than for calculating the impact of pruning various edges (Delle Fave et al., 2012b; Farinelli et al., 2014; Rogers et al., 2011). As such they have been used successfully in versions of the task-allocation problem (Farinelli et al., 2008; Macarthur et al., 2011; Ramchurn et al., 2015). As a result we consider this implementation (so called "fast") max-sum a suitable candidate for our work. This is shown in our inclusion of fast max-sum for task allocation in Chapter 3.

With a framework in place for allocating tasks, we now examine methods for predicting in advance where to place UAVs so that when task allocation occurs, the time for task attendance is minimised further. To this end, we explore algorithms to predictively place UAVs according to a belief map.

2.2 Predictive Placement Algorithms

Predictive placement is broadly described as the process of calculating some optimal distribution of objects in a space, according to some scenario-specific constraints or requirements. In our work, we consider algorithms that drive UAVs towards positions that minimise the time for them to attend incidents once the incidents are discovered at a future point. As a result, the task is predictive. The positions must be informed by any

available prior knowledge (Criterion 2)—in our scenario, a belief map—and should also consider spacing the UAVs out over the area as well since, in our scenario, we consider that each incident only needs to be attended by a single UAV: for multiple incidents it is therefore beneficial to disperse the UAVs over the area, in order to reduce incident response time. Intuitively, striking a balance between these two elements is important. Consider that with too much emphasis on placing UAVs close to areas of high incident likelihood the UAVs will "clump" together; reducing their effectiveness in responding to tasks that do appear in other regions. On the other hand, if the inter-UAV spacing is solely considered as the driver of the algorithm, the usefulness of the belief map will be lost. We therefore examine the literature for solutions to this problem.

While work has been done to determine placement to, for example, maximise communication coverage (Raissi-Dehkordi et al., 2004) or sensory information (Lin and Chiu, 2005; Sommerlade and Reid, 2010) (either using UAVs or other more general agents), no work has explicitly tackled the problem of placing UAVs in order to minimise future assignment of those UAVs to discovered incidents. Nonetheless, we draw inspiration from the work on sensor placement of Lin and Chiu (2005) and Bohm and Sawodny (2015), and consider the nature of the problem at a fundamental level and draw parallels with work in physics on minimum energy configurations (Jagla, 1999) for the similarities in the end goal (as discussed below). We will first discuss the nature of these problems before returning to their applicability to the UAV placement problem.

Minimum energy configuration problems typically consist of finding a spatial arrangement of particles in such a way as to minimise some global energy function (which we treat as an objective function to be optimised) that is a product both of the space and the relative positions of the particles. An early paper on the topic written by Wille and Vennik (1985a) details the problem as seeking to arrange particles in a space to minimise the sum of the interactions between them, when the interactions between any two points are known. They conclude that such problems fall in the category of NP-hard problems and note the importance of using robust heuristic methods to solve them, and in particular the *simulated annealing* approach introduced by Kirkpatrick (1984). Simulated annealing is an extended form of the Metropolis algorithm[2] used to determine energy-minimal configurations while avoiding local minima, which is highly desirable in our work. We also consider this algorithm superior to other methods such as hill-climbing or basin-hopping for its stochastic ability to quickly sample large configuration spaces (such as in extensive disaster spaces), and also for its high quality results with minimal computing time and overheads (Ingber, 1993). We discuss these further in the paragraphs below.

The form of the simulated annealing algorithm operating on a generalised state P is shown in Algorithm 1 and is described below.

[2]This is itself a particular form of a Markov chain Monte Carlo method developed for use in computational physics by Metropolis et al. (1953)

Algorithm 1 Simulated Annealing algorithm

//Initial state, threshold energy value, and initial temperature as input parameters//
CalculateOptimalConfiguration(P, *threshold*, T_{start})

1. $T \leftarrow T_{start}$
2. $E \leftarrow$`Energy_value(`P`)`
3. **while** E **is above** *threshold*
4. //State P changed by a small amount//
5. $P_{new} \leftarrow$`NewRandomState(`P`)`
6. $E_{new} \leftarrow$`Energy_value(`P_{new}`)`
7. //If the change lowers the energy, accept it//
8. **if** $E_{new} < E$
9. $P \leftarrow P_{new}$
10. $E \leftarrow E_{new}$
11. //If the change raises the energy, accept it with probability related to temperature parameter and change in energy//
12. **else**
13. probability$\leftarrow$`Acceptance_function(`$E_{new} - E$`,`T`)`
14. **if** probability $>$ `RandomFloatBetween(0,1)`
15. $P \leftarrow P_{new}$
16. $E \leftarrow E_{new}$
17. //Reduce temperature according to cooling schedule//
18. $T \leftarrow$`Cooling_schedule(`T`)`
19. **Return** P

The input parameters of the algorithm require (as well as the original state P) a temperature value T_{start} (and subsequent temperature T) and some threshold energy to ensure termination, although other methods of termination can also be used such as a fixed number of iterations or minimal change between successive system energies. At each step of the algorithm, the state P is changed by a small amount yielding a new energy for the system (Line 5). If this new energy is less than that calculated previously, the change is accepted and the new state is maintained for the next iteration. Otherwise, the change is accepted with a probability given as a function of the temperature parameter and the difference in energies (Line 13); in the original paper this was set as $\exp(-\Delta E/k_b T)$, where ΔE is the difference in energy levels, and k_b is a constant. It is important that this probability reflects that, as the temperature is reduced and the system "cools", changes allowing energy to increase are accepted with less frequency. The intuitive result is that at the initialisation of the algorithm the state is allowed to fluctuate almost freely between increased and decreased total energy, but that as the iterations continue the system increasingly favours decreases in energy. This enables the system to avoid becoming trapped in local minima while moving towards a global minimum value, which is ideally when the algorithm terminates. We note here that the terms energy and temperature– while representative of the thermal physics from which the method is derived– are both abstract variables which can be tailored to the specific goals of the function's implementation: in our context, deploying UAVs close to predicted positions

of tasks while maintaining separation between vehicles.

Given this background, Wille and Vennik (1985b) apply this algorithm to solve the problem of distributing charged particles in a space in order to minimise total energy. In this example, the energy is a function of the relative positions of the particles, since in a physical situation they exert repulsive forces on each other inversely related to their separation. This particular application provides the first element of a framework for using simulated annealing for UAV placement: namely that the method can be used to calculate a configuration for maximum separation between particles in the system. As stated above, the simple act of spreading UAVs over a search area fails to account for belief data. Thankfully the nature of the energy function does not specifically affect the algorithm's performance, and a global potential function reflecting a probability distribution such as a belief map could easily be integrated into the method. Indeed, work by Baras and Tan (2004) uses such functions to control swarms of simulated autonomous vehicles to deploy into different configurations in a discretised space, and Bohm and Sawodny (2015) use the algorithm for solving a sensor placement problem. The focus of the authors is to show that careful energy function selection can produce desirable arrangements and behaviour by using simulated annealing. As a result of this work, we believe that simulated annealing with an energy function depending on inter-UAV spacing and the belief map data (as per Criterion 2) is a viable solution for the prior placement of UAVs over a space in order to allow quicker attendance of incidents. This is achieved through careful tailoring of the algorithm's cooling schedule and the repulsive function operating to ensure good UAV spacing, and consideration of the exact use of the belief map to attract UAVs towards areas of high incident—and therefore task—probability. In other words, we use simulated annealing as-is, but with tailored heuristics for our required goals of reducing UAV travel time to tasks.

2.3 Path Planning and Exploration of a Search Space

Path planning is broadly defined as identifying a path in a robot's *configuration-space* in order to meet some conditions, such as ensuring that there are no collisions between robots and obstacles or other robots, and doing so in a computationally efficient way (Laumond et al., 1998). Here, a configuration-space can be thought of as a space containing any available configuration of a robot: be that position, orientation, or any other degrees of freedom such as the positioning of an appendage. The subject of the plan might be of part of the robot, such as a robotic arm, or the entire robot if it is a vehicle. In our context we are interested in the latter category since UAV position and movement is the main aim of an explorative path. Furthermore, we note that (unlike much of the literature) collision avoidance is not considered in our work since we deal with spaces vastly larger than the size of a UAV (Section 4.3.2) and the chance of two coming into contact is negligible. Consequently, we are faced with a broad field of algorithms that

calculate paths through space. In what follows, we focus on prior work on path planning that increases the likelihood of an exploratory UAV moving towards areas likely to contain people in need of assistance. More specifically, we consider situations where these paths can be informed by belief maps of the type described in Criterion 2.

As argued in Goerzen et al. (2009), there is no single path planning algorithm that suits every situation. Instead, specific problems require a specifically chosen path planning solution since each option comes with its own set of restrictions, such as computational overheads arising from the examination of the environment (Adler et al., 2014) or ability to plan over a belief map (Waharte and Trigoni, 2010). In general, they find that exact solution calculation[3] is often intractable due to the size and complexity of the search space. Moreover, providing guarantees of completeness or optimality with heuristic approaches can be challenging. Given this, the authors note that the vast majority of literature dedicated to the path planning problem is concerned with directing a robot towards some goal position (Khatib, 1986; Kuffner and LaValle, 2000). This represents a small subset of actual control problems and indeed, is not our primary concern here. Instead, as outlined in Section 1.1, we are concerned with exploration of a space, which does not necessarily require the termination of path planning at a specific point. Nonetheless, we evaluate four broad categories of the most common algorithms—encompassing the state of the art— with this in mind, and evaluate whether they could be adapted to our needs:

1. **Coverage Algorithms:** Algorithms designed to produce paths that cover a search space.
2. **Potential-Based Path Planning:** Algorithms that use simulated fields to repel robots from obstacles while driving them towards a goal.
3. **Rapidly-Exploring Random Trees:** Algorithms that produce simulated "trees" (of possible paths in a space) very rapidly, from which a path can be selected.
4. **Markov Decision Processes:** Algorithms designed to solve problems formulated as Markov Decision Processes. Specifically this encompasses Monte-Carlo Tree Search planning.

2.3.1 Coverage Algorithms

Coverage algorithms are those concerned with plotting routes through a space in order for a (typically sensor-equipped) robot to cover the entire area. While they benefit from being trivially simple to implement and are good for covering a large area comprehensively, their primary goal is inconsistent with Criterion 2 outlined previously; namely that

[3]That is, paths that are optimal according to the problem statement.

a path through the space should be informed by a belief map. In contrast, coverage algorithms tend to be un-directed once implemented, and simply seek to "cover" the search space without regard to any prior knowledge; typically using a striped "lawnmower" style pattern (Araujo et al., 2013; Li et al., 2011). In order to attend to incidents as quickly as possible, it is undesirable to visit the entire space except for the extreme case where there is no available belief data. While we wish to exploit belief data in order to uncover incidents more quickly, coverage algorithms are purely interested in allowing robots to traverse an entire area, and usually seek to minimise any re-visiting of previously visited areas (Gabriely and Rimon, 2002; Wong and MacDonald, 2003).

In benchmarking performance for our exploration algorithms in Chapters 3 and 4 we show clearly that a simple "sweep search" pattern—typical of coverage—is unsuitable for large scale exploration of a space, exactly because areas of high likelihood of containing survivors in danger are not prioritised.

2.3.2 Potential-Based Path Planning

Potential field methods are a class of planning algorithms which are based on equations from classical physics, introduced by Khatib (1986). In a path-planning context, a vehicle is modelled as a particle responding to forces acting on it as a result of a potential function assigned to the space in which it operates. A goal is represented by an attractive (negative) potential, and obstacles as repulsive (positive) ones. Koren and Borenstein (1991) present an explanation of the simplest form of such a model—known as *Virtual Force Field,* or VFF—along with a critique of its restrictions and limitations.

In more detail, a VFF is formed by modelling the environment either as a discrete arrangement of cells, or as a continuous vector field, where each position contributes—via a predefined function—to an overall vector "force" if it is occupied by an obstacle. Simultaneously, another component of "force" is directed towards a goal position. The combination of the two vectors is summed, and used to inform the steering rate of the robot controller. The overall effect of the components is to drive the vehicle away from obstacles whilst still tending towards the final goal position. Typically the area considered for repulsive forces to act on the vehicle is constrained to reflect the limiting sensing range of the agents.

This algorithm is designed for the direction of a UAV to some overall optimum location represented by a global minimum in the potential. This introduces a limitation for our purposes: in order to adapt it for use over a belief map—a situation with no obstacles or final goal—the algorithm would have to be implemented in a way that attracts the explorer to areas of high belief. Furthermore, if the map was decomposed into discrete components for computation on a cellular basis, the computation of the overall force

would become expensive as the sum would need to be performed over the entire disaster area or at least the area constrained by the robot's sensors. Moreover, Koren and Borenstein (1991) identify four crucial limitations to this general approach, which are described by the following resulting undesirable behaviour:

1. **Trap situations where local minima occur:** stable local minima in the potential function would result in the robot receiving no overall force value. This causes it to become trapped at some location other than its desired goal, or causing it to continually cycle or oscillate between several locations without any further progress to the goal.

2. **No passage between closely spaced objects:** objects which are close, but should nonetheless allow passage of the robot, can in practice result in overall force components which dominate that of the goal. This can result in the robot moving away from the gap, even though it would be able to pass safely through it.

3. **Oscillations in the presence of obstacles:** certain configurations of obstacles can result in stable oscillations between two positions, effectively trapping the robot in a small area.

4. **Oscillations in narrow passages:** any slight deviation from a straight trajectory down a narrow passage can result in oscillations from the continual change in force from the two walls. These can be unstable, and result in collisions with objects or other vehicles.

Limitation 1 is the most concerning for an application in disaster relief such as ours (where obstacle avoidance is beyond the scope of our work): local minima in the global potential function can cause behaviour which traps the robot in certain positions. Consequently, the exploration aspect (Criterion 2) of the project goals would not be satisfied, since exploration necessarily excludes a repetitive cyclical path in a localised region of a space.

As a result, work has been done to overcome local minima encountered in the potential field. A common example is to use a Brownian random walk to search for an escape from the minimum. To this end, Caselli et al. (2001) discuss this problem and the inefficiencies arising from using random walks, and instead suggest a method known as *StraightLineSelect*. This process generates straight sections of paths in random directions, and is shown to escape from a minimum more quickly than a Brownian solution. Nonetheless, as with the VFF, the focus remains on driving the path planner towards some predefined goal. We can envisage ways to modify the algorithm to better meet our requirements; for example by designing it to continually seek out different points of minimal potential (specifically areas of the belief map with higher probability). However, this fails to account for the explorative value of the adjoining paths. In more detail, Criterion 2—detailing exploration—is more nuanced than simply moving from one point of

"high" likelihood to another, which would neglect areas of low (but non-zero) probability of incident detection. Further limitations of such an approach detailed in Sujit and Saripalli (2014) note that as well as the above, VFF methods generally require careful tuning of multiple parameters to create desirable UAV behaviour: an impractical overhead to consider in real deployment scenarios. In this sense, potential field based methods do not satisfy our requirements.

2.3.3 Rapidly-Exploring Random Trees

Unlike discrete-space searches such as those detailed above, Rapidly-Exploring (or expanding) Random Trees (RRTs) are a versatile class of space searching algorithm developed by LaValle (1998) for application to path planning over either discrete or continuous space. The basis of this algorithm is the rapid extension of a tree of connected nodes into a space, by growing the tree towards large, previously unexplored, areas. Once a tree has been grown sufficiently, or reaches a predetermined goal, a path through the tree can be selected easily by either graph search methods, or simply tracing back along the edges from the goal. The specific advantage of RRTs over other tree search algorithms is its ability to spread extremely rapidly over an area, with relatively minimal computational cost. Nonetheless, there exists problems with growing RRTs in a coordinated fashion between multiple UAVs; their random growth patterns and the lack of simple utility values between multiple paths mean synchronising and coordinating future actions is unfeasible without incurring impractical overheads (Kumar and Michael, 2012) or centralisation (such as by sharing entire random tree-growth among all agents).

In more detail, the RRT algorithm selects random points in space and expands the nearest tree-node towards that point by a fixed amount. In this way, the algorithm efficiently selects points in space which have not been previously explored: consider that a newly created point will fall in the Voronoi region[4] of the node which is subsequently expanded (by virtue of the nearest-node condition). The "frontier" points of the tree will clearly have larger Voronoi regions and therefore be expanded more frequently, continuously driving the tree outwards. While the trees need not necessarily represent physical paths through a space (and could instead represent configuration arrangements of part of the robot), this is the context in which they prove useful to the work outlined in this report.

An example construction procedure is given by Algorithm 2 for generating a tree T with K vertices, with a graphic of a tree growing shown in Figure 2.2. The parameter u describes the velocity dynamics of the growth over an interval Δt: in practice this dictates how far towards a newly selected point the tree can grow. From an initial position x_{init}, a new position is picked in the state-space (Line 4) and its nearest neighbour, x_{near}, in

[4]The Voronoi region of a node in a space is defined as the locus of points which are closer to that node, than any other node (Aurenhammer, 1991).

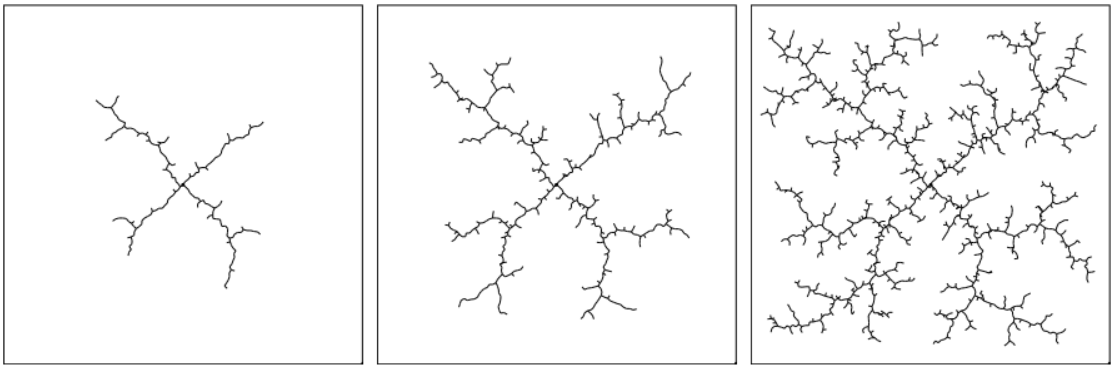


FIGURE 2.2: RRT expansion from an initial point in the centre of the figure (LaValle, 1998).

Algorithm 2 Standard RRT Growth

//Algorithm initialised with initial position or path, x_{init}, the final number of vertices K, and growth interval Δt//
Generate_Tree(x_{init}, K, Δt)

1. T.`init`(x_{init})
2. **for** k=1 **to** K
3. //Picks a new random state in the space//
4. x_{rand}←`Random_State()`
5. //Finds nearest existing node//
6. x_{near} ←`Nearest_Neighbour`(x_{rand}, T)
7. //Describes how near to the state the tree can expand in interval Δt//
8. u←`Select_Input`(x_{rand}, x_{near})
9. //Creates new node directed towards x_{rand} by some amount governed by u//
10. x_{new} ←`New_State`($x_{near}, u, \Delta t$)
11. T.`add_vertex`(x_{new})
12. T.`add_edge`(x_{near}, x_{new}, u)
13. **Return** T

the tree is located (Line 6). A new node for the tree is created in the direction of the new position, a fixed distance from x_{near} (Lines 8 to 12).

Since LaValle's original work, many attempts have been made to adapt the RRT method to more realistic planning scenarios. These include dealing with uncertainty in agent and obstacle position (Bry and Roy, 2011), path planning with a focus on online replanning (Zhen et al., 2014; Kothari and Postlethwaite, 2012), and for finding routes towards a goal while collecting information about the environment or tracking a target (Alighanbari et al., 2006; Levine et al., 2012; Xinggang et al., 2014).

A limitation of most RRT approaches, which, shared with the other approaches outlined in this chapter, is that because it is designed to spread quickly and widely over an area, it favours problems with a goal of reaching a specific position in the search space (by quickly establishing different routes towards it). Indeed, even the extensions mentioned above that attempt to address the information gain along a path do so within the context

of a specific final goal. This problem persists in work conducted for coordination of RRT based planners such as in Cap et al. (2013), which, as previously mentioned, is already a difficult task in RRT fields. In the context of disaster space search and rescue, it is more instructive to consider planners that work towards constructing exploratory paths which tend towards areas of likely incidents, but which are not focussed upon terminating at a specific point. Efforts towards such a formulation have been made by Scerri et al. (2008), who introduce a modified planner which extends outwards from promising nodes in the tree, rather than towards preselected locations in the space. The chosen metric for identifying the most useful paths is a dynamic information-entropy map, similar in function to the belief data condition (Criterion 2) of the goals outlined in Section 1.1, which is obtained via a logarithmic transformation of target probability (in our work, this would represent incident detection probability). The use of an entropy map rather than a belief map, is due to the author's goal to minimise uncertainty in the tracking of a target, rather than deliberately attending areas of high detection probability as we seek to do. As well as using entropy data to inform the growth of the tree itself, the modified RRT planner also removes the most computationally expensive part of the traditional RRT method, the nearest-neighbour computation.

This modified RRT is detailed in Algorithm 3, and works as follows. A list of nodes is kept sorted according to each node's *priority* (Line 12) as calculated from a combination of the *cost* associated with its position coordinates (generated from the entropy map, which returns values representing the numerical cost of moving to locations), and the number of times it has been previously expanded. Nodes with a high cost and which have been expanded many times previously are considered to have low priority for future expansion. The node with the highest priority is expanded outwards ten[5] times to create ten new neighbours (Line 15), which are then added to the priority list. The node incurring the lowest associated cost is designated as the best choice (Line 24), and a path can be obtained by iterating back over its *prev* set after sufficient expansion.

Scerri et al. also describe a way to consider the re-planning problem: by planning short paths the robots may fail to move towards high value areas too far away; however, planning long paths reduces reactivity to new information in the map (i.e. from dynamic belief maps). The solution offered is to plan long paths, but only carry out a short portion of the movement before recalculating. High value areas can be reached by repeatedly following part of a longer path to that area, but reactivity is maintained due to the more frequent replanning schedule.

For these reasons, we consider RRTs suitable for our work in allowing an explorative UAV to quickly plan and explore a disaster space. We discuss this again in depth in Section 3.2.2, and outline the utility function we use to determine growth of the modified RRT detailed above.

[5]This is a parameter chosen by the author because it performs well empirically.

Algorithm 3 Modified RRT growth

RRTPlanner(x, y, CostMaps, time,state)

```
1. plist←∅
2. n←⟨x, y, cost = 0, prev = ∅, priority = 0⟩
3. //Priority value calculated from cost, and number of previous expansions of n//
4. n←ComputePriority(n)
5. plist.insert(n)
6. best=n
7. foreach 20000
8.     //Highest value node chosen for expansion//
9.     n←plist.removeFirst()
10.    n←ComputePriority(n)
11.    //Re-inserted with new priority, so that plist keeps ordering//
12.    plist.insert(n)
13.    foreach 10
14.        //Generates new node a small distance from n//
15.        n'←Expand(n)
16.        n'.prev←n
17.        //Cost of new node calculated from map//
18.        n'.cost←Cost(n',CostMaps)
19.        n'.priority←ComputePriority(n')
20.        //n′ inserted in order into plist//
21.        plist.insert(n')
22.        //Keeps track of best node//
23.        if n'.cost<best.cost
24.            best←n'
25. Return best
```

2.3.4 Markov Decision Processes

The formulation of the explorative path-planning problem as a *Markov decision process* or MDP presents a useful baseline from which to construct a coordinated path planner. Specifically, the decomposition of a search space into discrete cells also discretises the action space resulting from movement through the cell. This constrains the size of joint-action search spaces of the type considered when coordinating multiple UAV paths simultaneously, and could therefore overcome some of the problems highlighted in the literature about planning multiple paths, although problems of optimality will of course persist (Feyzabadi and Carpin, 2014). MDP methods also carry the benefit of incorporating future-planning, which we can use in situations where the scenario model evolves with time (as we do, after discussing this in Section 4.1).

Formally, an MDP is defined as a tuple $\langle S, A, R, P \rangle$, where S is a finite (except in the case outlined in Section 2.3.4.2) set of states, A is a set of actions to be performed by the agent(s), R is a reward function: $R : S \times A \to \mathbb{R}$ (for instance, $R(s, a)$ represents the reward from state s after taking action a), and P is a model of transition probabilities,

such that $P(s' \mid s, a)$ gives the probability of transition from state s to s' given an action a (Guestrin et al., 2001). These terms are used to construct the *action value function* Q, describing the effective reward of a given action and possible future actions over a given time horizon, according to the Bellman Optimality Equation (Sutton and Barto, 2000):

$$Q(a, s) = R(a, s) + \gamma \sum_{s'} P\left(s' \mid a, s\right) \max_{a'} Q\left(a', s'\right) \tag{2.5}$$

Here, the sum is performed over $P(s' \mid a, s) \max_{a'} Q(a', s')$; the product of the probability of the system changing to the state s' given that an action a has been carried out on the current state s, and the optimal action value function for the new state, given the preferred action is then carried out. The term $\gamma \in (0, 1)$ is a discount factor that places more value on immediate rewards relative to future rewards, and ensures Q has a finite value when computed over infinite time horizons.

In such a formulation, simple state-spaces (particularly those with a small number of states which cycle between each other) are frequently represented by Bayesian Networks—graphs of states as nodes with transition probabilities as edges (Ben-Gal, 2007)—and work exists which seeks to solve these relatively restrictive examples (including in a coordinated, multi-agent fashion (Adlakha et al., 2008; Guestrin et al., 2001)). Conversely, more complex environments with potentially infinite state spaces are frequently constructed to resemble trees, with each node representing a state and each edge a set of joint actions that result in the relevant next state according to the probability function P. This approach results in solution formulations amounting to efficient traversal algorithms—specifically, of considering the best series of actions through the tree such that each state visited maximises Q— with the benefit over more simplistic and naïve constructions (such as those mentioned above) that complex sets of diminishing returns can be included and modelled with relative ease.

In Chapter 4 we outline an extension to the exploration problem which calls for a broader attention to areas of "danger" than simply identifying localised incidents (specifically one involving time dependent considerations of survivability); whilst allowing multiple explorative UAVs to explore together—a scenario ideally suited to an MDP formulation. Specifically, we build on work using *Monte-Carlo Tree Searches* ((Browne et al., 2012), see below in Section 2.3.4.1) to traverse a tree representing possible UAV actions. The *Upper Confidence-Bound for Trees* (UCT) Monte Carlo Tree Search traversal algorithm developed by Kocsis and Szepesvari (2006) is particularly promising for our needs and is outlined in the following subsection.

Subsequently, to account for possible sensor uncertainty in an environment, another method recently applied to discrete search representations of belief maps is a *partially observable Markov decision process*, or POMDP. POMDPs (introduced in Smallwood and Sondik (1973)) differ from the MDP method by including in the planning stage the

consideration that a given observation o performed by the agent may not necessarily record the actual state s of the environment: specifically accounting for sensor uncertainty. In our context, this is most often applied to path-planning (although some work exists using POMDPs to plan task allocation, such as in Spaan et al. (2011)). However, continuous space POMDPs are frequently regarded as too computationally expensive to be tractable as path planners (Huang and Gupta, 2008), and although work has been done to create discrete-space versions by using random sampling (Kurniawati et al., 2008), these are still expensive in their current form when compared to the other algorithms available to us. Principally, we consider POMDPs unnecessary to our scenario because of particular assumptions made in our reformulated problem (Section 4.1) about the nature of an observation and its impact on further planning: specifically that future planning in our problem does not depend on the nature of an observation o at a particular time. Further detail is provided in the formulation in Section 4.1.

We now return to our interest in MDPs, and consider the benefits of a UCT Monte-Carlo Tree Search (MCTS) and whether it forms a suitable candidate for our applications.

2.3.4.1 UCT Monte Carlo Tree-Search Method

As already indicated, the "tree" to be searched by a MCTS method is constructed of nodes representing possible state spaces, with edges representing the joint actions taken to reach that state. In this respect, they are functionally very similar to trees used in game or decision theory; a well studied field (Browne et al., 2012; Coulom, 2006; Gelly and Silver, 2011; Chaslot et al., 2008). Much research exists considering the various methods of using such a tree to plan a series of actions, from simple depth/breadth first searches to more nuanced algorithms that exploit the features of a tree to better judge where to "explore" (that is, simulate by expanding the tree to new action nodes) next. Naïve searches typically fall victim to the high branching factor endemic to trees with large joint action spaces; being of $O\left(\mid A \mid^N\right)$ where N is number of agents and $\mid A \mid$ is the size of the action space for each agent. As a result, Monte-Carlo Tree Searches offer a principled expansion method that seeks to balance the value of "exploring" new sequences of actions (effectively broadening the tree) and "exploiting" existing action sequences to examine them over additional future states (deepening the tree).

The typical structure of general MCTS methods involves the expansion of the action tree according to a four step structure (which we explain in detail below) of exploring, expanding, rollout, and backpropagation. The basic format is outlined in Algorithm 4, and expanded on hereafter. Below, we outline the specifics of the UCT form of this algorithm (Browne et al., 2012), regarded as the state of the art algorithm for growing and solving MDP trees.

Algorithm 4 Tree Growth

//Basic tree growth. Returns newly updated tree//

1. $node \leftarrow$Choosenode($rootnode$)
2. $Newnode \leftarrow$Expand($node$)
3. $Newnode.value \leftarrow$Rollout($Newnode$, direction ($node, Newnode$))
4. Backpropagate($node, Newnode$)

Algorithm 5 Node Expansion Selection

//Chooses node to expand next//
$node \leftarrow Rootnode$
Choosenode($node$)

1. **if** $node.expansions < maxexpansions$ **then**
2. //Increment number of expansions for that node//
3. $\mathrm{N}_e\,(node) \leftarrow \mathrm{N}_e\,(node) + 1$
4. Return($node$)
5. **else**
6. //UCT condition for choosing to explore new action//
7. $nextnode \leftarrow \arg\max_{child \in children} child.value + c\sqrt{\frac{2\ln \mathrm{N}_e(node)}{\mathrm{N}_e(child)}}$
8. //Continues iteratively until reaching node without full expansion//
9. Choosenode($nextnode$)

The initial selection of a tree node for expansion (Line 1) is outlined in Algorithm 5. In more detail, the algorithm iteratively explores down the tree through the nodes, returning as its highest priority any nodes which have not had their full compliment of children expanded from it (Line 4). Otherwise, the child node that maximises the UCT condition (Line 7) is returned and the process is repeated iteratively. Here, the argument-maximum function is performed over all children nodes, and takes as arguments the number of previous expansions of the child or parent (function N_e), the calculated estimated reward for the child node (*child.value*, explained below), and a weighting factor between the first (exploitation) term and the second (exploration) term c, usually set empirically (Kocsis and Szepesvari, 2006; Guez et al., 2012). In this way, the algorithm continually selects a trade-off between nodes with a high average value (i.e. a high *node.value* term) and those which have yet to be explored.

The function in Line 2 in Algorithm 5 entitled "Expand" entails the creation of a new child node to the variable *node* returned in Line 1, typically by selecting a random new available action from that node, that has not previously been selected. The new action node is added to the tree as $Newnode$, and an estimate of its reward is calculated both from its immediate contribution to the Q (from Equation 2.5) and an estimate of future reward via a "rollout".

To elucidate somewhat the function of the *node.value* property; this value represents an

Algorithm 6 Backpropagation

Backpropagate($node, child$)

1. $oldvalue \leftarrow node.value$
2. //Basic implementation of cumulative average formula//
3. $newvalue \leftarrow oldvalue + \frac{(child.value - oldvalue)}{\mathrm{N}_e(node)+1}$
4. $node.value \leftarrow newvalue$
5. **if** $node.parent \neq$ **none then**
6. Backpropagate($node.parent, node$)

average of a node's immediate reward and the predicted future reward returned from the Rollout function. This function—mentioned in Line 3 (Algorithm 4)—is not defined explicitly in the original literature, but is representative of a coarse estimation function to return some value of future actions from the selected node: for example, by performing a set number of further actions in a random fashion and returning the cumulative (or averaged) reward, to be averaged with the existing *node.value*. Typically the rollout stage is not discussed in depth in papers on the subject—since in principle it is simply a coarse estimate of future reward—although in Chapter 4 we outline and justify the explicit form of our rollout function.

The backpropagation stage of MCTS (Line 4 in Algorithm 4) is outlined in Algorithm 6, starting with a leaf node and passing values back up the tree until it reaches the root. At each iteration, the value assigned to the node is modified using a moving-average to account for the average of the values of those nodes upstream. This reflects the combination of rewards obtained from Equation 4.6 as being due to a node's immediate reward and the sum of future rewards further down the tree.

Against this background, we find useful elements of this class of algorithm for our problem of coordinating multiple exploratory UAVs; specifically where locality of agents can be used to reduce calculation overheads. In particular, the generality of MDP formulations lends itself well to the construction of a simulation environment given numerical data used for a belief map, as well as having a number of well-established solutions. More specifically, work by Amato and Oliehoek (2015) utilises factored tree-searches for partially observable MDP solutions; exploiting problem structure to allow factorisation in a way reflective of local state spaces and interactions. This is a technique we apply to disaster-scenario data in our contributions (see Section 4.2.1 for further details).

In summary, the explicit formulation of the impact of future rewards on current planning and the balance between exploration and expansion of the tree in UCT proves ideal for solving path planning within the formulation of a temporal "danger" over a disaster area, discussed in Chapter 4. Furthermore, the method also allows for trees representing arbitrary sets of actions—not just for a single agent—that satisfy our requirements for

coordinating multiple UAVs simultaneously. As a result it is a form of coordinated MCTS that we implement in our work.

2.3.4.2 Continuous Space Monte Carlo Tree-Search

We note that, thus far in our discussion, the use of MCTS methods appears to be restricted to discrete action spaces. Specifically the selection and growth of action nodes in the tree rely on there being a clearly defined point where a node in the tree is "fully" expanded. In cases with a continuous (and therefore infinite) action space, this requirement cannot be met. Instead, methods must find ways to sample the available search-space for suitable representative actions whilst also continuing to balance the tree-search problem of exploring down the tree or branching across the tree. While some work simply deals with discretising a continuous space in a principled way (Broeck and Driessens, 2011), we are instead interested in the direct application of continuous data (or at the very least, arbitrary datasets that might not be consistently discretised). Specifically this addresses Criterion 2 in Section 1.1, requiring the use of belief data that need not be available in a discrete form. As such, we wish to ensure a path-planning algorithm capable of operating in a (more realistic) continuous domain.

Against this background we find parallels between this problem and the so-called *many* (or *multi*) *armed bandit* problem of game theory (Couëtoux et al., 2011b). This is a planning scenario where the optimal action must be drawn from a variety of "one armed bandit" machines with (unknown) probability distributions over their rewards. Although dissimilar to our exploration problem in terms of temporal planning, work in this domain has (for some time) sought to address the case of a continuous set of "bandits" (Agrawal, 1995), or the very similar situation of an infinite number of discrete "bandits" (Wang et al., 2008). The latter of these works introduces upper-confidence methods to the bandit problem. More recent research has focussed on beginning to apply UCT methods to continuous tree domains (Rolet et al., 2009; Weinstein and Littman, 2012), and there has been some success in applying it to specific problem domains; with particular attention given to how to choose which of the continuous domain of actions to sample in the tree (Auger et al., 2013; Couëtoux et al., 2011a; Couëtoux, 2013; de Waard, 2016). In particular, we note the promising results of the so-called Polynomial Upper-Confidence Trees (PUCT) algorithm introduced by Auger et al. (2013) that shows improvements over a similar earlier adaptation of a continuous armed bandit situation—Hierarchical Open-Loop Optimistic Planning (HOLOP) (Weinstein and Littman, 2012)—both of which present methods of selecting further actions from which to generate new nodes in the search tree; and heuristics to determine when to "explore" and when to "exploit" nodes. Another welcome feature of PUCT is that it requires only a "black box" sampler of future actions and the sample rewards expected.

The principal contribution of PUCT is the condition of expanding a node: namely to replace the condition in Line 1, Algorithm 5. The condition to expand a node is instead given by:

$$\textbf{if } N_e(z)^{\alpha} > N_e(z-1)^{\alpha}$$

Here, the z argument denotes the action node[6] being considered, $N_e(z)$ the number of visits to the node during the current tree expansion, and α is a variable (known as the *progressive widening constant*) dependant on the depth d of the node in the tree given as $(10(d_{max} - d) - 3)^{-1}$—with d_{max} denoting the final depth in the tree search (i.e. a constraint on the number of expansions). In this case a balance is struck between expanding more widely further down the tree (with increased depth) against the number of times higher up nodes are visited (i.e. those closest to the root get expanded and updated more often). We note that despite the performance gains over HOLOP, the authors note concerns over cyclicality between states. However in a sequential exploration problem where observations permanently affect the state of the environment (as is the case in UAV disaster exploration scenarios) such concerns are irrelevant. Furthermore, the authors explicitly note the applicability of their algorithm to general continuous sequential decision making problems: precisely the type of problem we seek to address. As such, we deemed the PUCT method with a progressive-widening parameter a candidate for the extension of the exploration problem into a continuous state-space domain. This enables us (in Chapter 5) to implement a continuous form of tree-search using the rewards and utility from our model of the disaster scenario.

2.4 Summary

In this chapter, we have introduced the state of the art in path planning, task allocation, and predictive placement research and selected work that is particularly relevant to the goals outlined in Section 1.1, while rigorously examining the quality and performance of various methods. Although each of the methods outlined have various merits and drawbacks, we note that no work has so far attempted to unite the issues of explorative path planning, task allocation, and prior placement of UAVs. We conclude that some methods with particular merit to the creation of a unified framework are:

- Path planning via the modified RRT approach presented by Scerri et al. (2008) for single-agent exploration. We regard this method as being the most promising for allowing fast exploration (vital in a disaster situation) while fully utilising any available belief-data.

[6]The original work distinguishes between transition (so-called "random") nodes and action (or "decision") nodes. For the purposes of this argument we focus on the implications of choosing actions at the decision stage, in line with our interests.

- Task allocation using a fast max-sum algorithm, because of its ability to allow fast empirically-optimal decentralised coordination.
- Calculation of predictive placement using a simulated annealing framework, to allow us to formulate a potential from belief data tailored to our needs of placing UAVs as close as possible to sites of future incident discovery.

We consider these separate components as operating together in the same scenario but performing very different roles; as well as having various metrics tailored to our exact specifications in order to prove useful in the disaster-response regime.

Thereafter, we identified the need for allowing path-planning of a space to occur between multiple coordinating UAVs, rather than from an individual. To that end, we regarded Monte Carlo Tree Search the best candidate class of algorithm. Primarily, this is on the basis of the step-wise growth of the tree that simultaneously enables future planning, but also flexibility in creating joint-action trees (for multiple UAVs simultaneously) or co-ordinated growth between trees where required (we discuss in detail our implementation in a discrete space in Section 4.2, and in a continuous space in Section 5.3).

In the subsequent chapters, we draw on the work outlined in this literature review to meet the requirements outlined in Chapter 1 for algorithms to coordinate UAVs in disaster response scenarios.

Chapter 3

The Path Planning and Task Allocation Mechanisms

Based on the analysis of the literature presented in Chapter 2, here we present a series of mechanisms for the planning of a path for incident discovery, the prior placement of UAVs to positions likely to be close to incidents, and the assignment of these UAVs to incidents as a task allocation process; as per the requirements and conditions outlined in Section 1.1.

In more detail, Section 3.1 begins by formalising the problem of discovering and responding to incidents in a disaster area. Section 3.2 then introduces our solution formalism, specifically the objective functions for path planning and task allocation. Sections 3.2.1, 3.2.3 and 3.2.2 then detail our chosen path planning, task allocation, and prior placement algorithms respectively. Next, Section 3.3 describes the form of our experiments to evaluate our algorithms. Then in Section 3.4 we present the empirical evaluation of our mechanism, showing its efficacy. Finally we conclude with a summary of the empirical results and further discussion on the performance of the mechanism in Section 3.5.

3.1 Problem Statement

As discussed in Chapter 1, discovering or verifying the location of *incidents* such as fires or collapsed buildings is of key importance to first responders in disaster relief situations, and it is of particular importance that a system be devised to perform this task autonomously. Here, we formulate this scenario as a joint path-planning and task allocation problem, where (referring to the discussion in Section 1.1) a fast moving *high-level* UAV is required to produce and follow a path over the disaster space to discover incidents,[1] then to create tasks for *low-level* UAVs that aim to attend these tasks in the

[1] In this chapter, we consider a single high-level explorer, however we expand this to multiple explorers in Chapters 4 and 5.

shortest total time possible. Henceforth, we refer to the problem outlined above as the Path Planning, Task Allocation and Placement problem (or PPTP problem).

More formally, we denote the set of UAVs as $\mathcal{U} = \mathcal{U}_h \cup \mathcal{U}_l$, where $\mathcal{U}_h$ is the set of high-level UAVs,[2] and $\mathcal{U}_l$ is the set of low-level, imagery-providing UAVs. The goal of a high-level UAV $u' \in \mathcal{U}_h$ is to explore a bounded 2D area $\mathcal{S} \subseteq \mathbb{R}^2$ defining the disaster space, where we denote individual locations as $\mathbf{l} \equiv (x, y) \in \mathcal{S}$.[3] In order to perform this exploration, the UAV calculates a path (or *trajectory*) T of ρ vector waypoints to follow $T = (\mathbf{l'}_1, \ldots, \mathbf{l'}_\rho) \subset \mathcal{S}$. While exploring $\mathcal{S}$, the high-level UAV aims to discover all incidents belonging to the set $I = \{i_1, \ldots, i_j\}$ (where we assume $j < \infty$), and it will continuously explore until it has done so. An incident is considered discovered when the UAV u' passes within a threshold distance of the incident, reflecting the restricted range of sensory equipment. More formally, given a position function for all UAVs in $\mathcal{U}$, and incidents, $L : I \cup \mathcal{U} \to \mathcal{S}$, we can compute the Euclidean distance between the u' at position[4] $\mathbf{l} \in \mathcal{S}$ and an incident i at position $L(i)$ as $\partial_i^{\mathbf{l}} = \|L(i) - \mathbf{l}\|$, where we use the norm of the difference between the positions $L(i)$ and $\boldsymbol{s}$ to denote the magnitude of their separation. Consequently, given a threshold $\delta \in \mathbb{R}^+$, we consider an incident discovered if:

$$\partial_i^{\mathbf{l}} \leq \delta \tag{3.1}$$

We assume here that the incidents are static (i.e. their positions do not vary with time—a sensible assumption where incidents represent survivors that are trapped). Furthermore, we assume that incident discovery according to Equation 3.1, and the positions returned by the function L, are deterministic.[5] Additionally, prior information on the position of these incidents is supplied in the form of a *belief map* describing the likelihood of incidents being at a given location, which informs the construction of the path T mentioned previously (see Section 3.2.2), in order to increase the likelihood of incident discovery by u'. Formally, we consider a function $\mathcal{M} : \mathcal{S} \to \mathbb{R}^+$, representing a scalar field such that:

$$\mathcal{M}(\mathbf{l}) \propto P(i)dxdy \tag{3.2}$$

where $\mathbf{l} \in \mathcal{S}$, and the function P denotes the probability of an incident being located within an interval $dxdy$ in $\mathcal{S}$. This environment construction can be seen in the first step of Figure 3.1.

Once incidents are discovered, they can be attended by members of the set of low-level UAVs $\mathcal{U}_l = \{u_1, \ldots, u_m\}$. The goal for the set of UAVs $\mathcal{U}_l$ is for them all to reach the

[2] We address the addition of more in Section 4.1.

[3] We denote any such position vectors in bold typeface, and use κ as a generic index when required.

[4] Note that the high-level UAV need not be positioned exactly on a waypoint in T, but could be travelling between them. Hence, we use $\mathbf{l}$ to denote position rather than $\mathbf{l'}$.

[5] I.e. we do not consider uncertainty at this stage. This is a sensible assumption with regards to positioning, since Global Positioning System uncertainty is negligible in comparison to the typical large distances encountered in disaster scenarios. Nonetheless, we address determinism implicitly in our re-formulation of the survivor discovery problem in Chapter 5.

location of the discovered incidents I in the shortest cumulative time t possible (i.e. the total time taken by all UAVs). Minimising the total travel time in this way is comparative to an allocation seeking to optimise battery consumption for UAVs while allowing them to move quickly to their respective tasks. In real disaster situations these considerations are vital to maintain a consistent presence of UAVs over an area. It should also be noted that incident attendance can of course commence before the time that all incidents have been discovered.

Now, to construct the objective function that minimises the total travel time, we define the *task* of allocating a given UAV $u \in \mathcal{U}_l$ to an incident $i \in I$ at $L(i)$ as $\tau^u_{L(i)}$, belonging to the set of all possible tasks $\mathfrak{T}$. We assume that the number of UAVs is equal to the number of incidents: $m = j$, meaning that each UAV $u \in \mathcal{U}_l$ can only be allocated one incident. This is a reasonable assumption for the scenario we outline, in which the explorer traverses the space quickly enough that the task allocation UAVs do not spend much time at their tasks before the exploration is complete. As such, this might reflect the initial allocation of a system in a disaster situation; and any rescheduling would constitute a separate problem to be solved thereafter. [6]

Consequently, our goal is to find an optimal set of allocations (we discuss the conditions of optimality below) $\boldsymbol{\tau} \subset \mathfrak{T}$where for any given pair $\tau^{u_\alpha}_{L(i)} \in \boldsymbol{\tau}$ and $\tau^{u_\beta}_{L(i')} \in \boldsymbol{\tau}$, $u_\alpha \neq u_\beta$ (where α and β are generic indices $\alpha, \beta \in \{1, \ldots, m\}$) and $i \neq i'$. Once a set of allocations $\boldsymbol{\tau}$ is chosen, and each UAV is assigned their task $\tau^u_{L(i)} \in \boldsymbol{\tau}$, the UAVs are required to move from their current position to the location of their allocated incidents, so that ultimately $\{L(u_1), \ldots, L(u_m)\} = \{L(i_1), \ldots, L(i_n)\}$ (we note that as these are not sequences, the ordering of the elements in each set do not matter). Given that the UAVs start at locations $\mathcal{L}_{start} = \{L(u_1)_{start}, \ldots, L(u_m)_{start}\}$ when receiving their allocations, some time must pass as they travel from their starting positions to the position of their allocated incidents. As a result, each allocation $\tau^u_{L(i)}$ maps to a travel time $t\left(\tau^u_{L(i)}\right)$, and each set of allocations $\boldsymbol{\tau}$ maps to a sum of the times for each individual allocation:

$$t(\boldsymbol{\tau}) = \sum_{\tau^u_{L(i)} \in \boldsymbol{\tau}} t\left(\tau^u_{L(i)}\right)$$

The time $t(\boldsymbol{\tau})$ is the value we seek to minimise by finding the optimal allocation $\boldsymbol{\tau}^* \subset \mathfrak{T}$ as follows:

$$\boldsymbol{\tau}^* = \arg\min_{\boldsymbol{\tau} \subset \mathfrak{T}} (t(\boldsymbol{\tau})) = \arg\min_{\boldsymbol{\tau} \subset \mathfrak{T}} \sum_{\tau^u_{L(i)} \in \boldsymbol{\tau}} t\left(\tau^u_{L(i)}\right) \tag{3.3}$$

[6] Future scheduling and reallocation has been studied in other contexts (Agmon et al., 2010; Shen and Salemi, 2002), but is beyond the scope of this work.

Note that we assume all low-level UAVs have the same constant velocity of $v \in \mathbb{R}^+$ and travel in straight lines to their designated incident.[7] As a result, the time for a UAV u to reach an incident i is now dependent on the distance between them, ∂_i^u, in the following way: $t\left(\tau^u_{L(i)}\right) = \frac{\partial_i^u}{v}$. Given this, we can see that the problem of reducing the total time spent moving to tasks can be recast as one to reduce the total distance travelled by all members of the set $\mathcal{U}_l$ to reach the locations of the incidents in I. Consequently, Equation 3.3 can be recast as:

$$\boldsymbol{\tau}^* = \arg\min_{\boldsymbol{\tau} \subset \mathfrak{T}} \sum_{\tau_i^u \in \boldsymbol{\tau}} \frac{\|L\,(i) - L\,(u)_{start}\|}{v} \tag{3.4}$$

where, as before, the norm indicates the magnitude of the distance between the incident i and the UAV u. The minimisation is then performed over the possible allocations $\boldsymbol{\tau} \subset \mathfrak{T}$ to find the one with the shortest total distance travelled by each UAV to an incident.

3.1.1 Spatial Correlation Modelling

We note that, throughout discussions of path planning—both in this chapter and in Sections 4.2 and 5.3—we will not be considering (explicitly) any correlation between observed incidents/locations and surrounding areas. In more detail, some planning literature (Singh et al., 2009; Stranders, 2010) considers the impact that sensing or observing one location has on surrounding locations: namely that if you detect something, the likelihood of further detections in that vicinity increases. We do not consider such correlations for two reasons.

Firstly, the data we observed from disaster scenarios (especially that discussed in Section 4.3.2 and shown in Figure 4.7) was seen to be extremely flocculent, and there was little apparent cohesion between likely locations of people in neighbouring cells. In other words, the only useful observations about certain spatial locations being more or less likely to contain incidents or survivors was already explicitly considered during the sampling stage of the path planning.

The second reason is that any modelling of spatial correlations between locations of survivors would need to be intrinsically based on real-world data. Existing research on population distributions demonstrates that modelling large distributions of people is extremely difficult (Deville et al., 2014; Patel et al., 2016; Wilson et al., 2016) because of the number of variables involved (for instance types of buildings, local geography, and baseline levels of population movement). We concluded that (although theoretically possible) such an undertaking was beyond the scope of our work, and—in any case—in

[7] The assumption of constant velocity is sensible for similar UAVs in the set $\mathcal{U}_l$ (since they are likely to be of similar type), and UAVs will generally travel in straight lines except where there are restrictions on flying zones.

a disaster scenario plans would need to be conducted quickly over readily available data (such as might be represented in the belief maps we represent) rather than relying on time consuming and complex anthropological models.

3.2 Solution Formulation

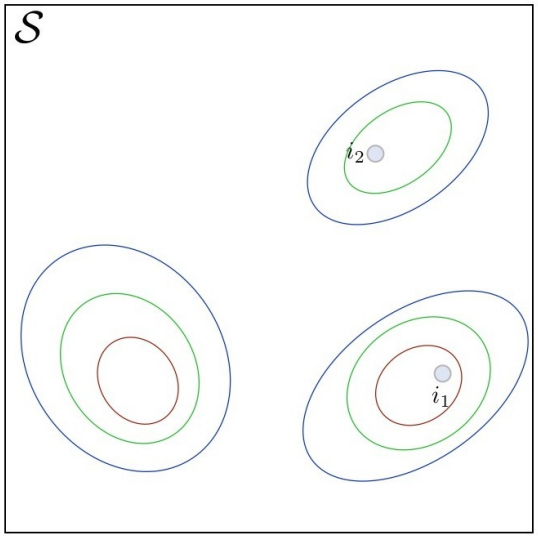
To address the PPTP problem, we now present a complete mechanism which, given a belief map of incident location over a search space, utilises a combination of heterogeneous UAVs to isolate possible incidents and provide imagery of the locations back to the first responders without the need for manual control. As discussed previously in Section 1.1, this belief map would be constructed from available crowd-sourced data in a real situation (such as from the Ushahidi project (Morrow et al., 2011) or Crisis Mappers (Crisis Mappers, 2013)). We consider the solution in three stages outlined below. We refer each of these algorithm stages to the step in the execution of the complete mechanism, shown graphically in Figure 3.1:

1. **Prior Placement:** Goal positions for $\mathcal{U}_l = (u_1, u_2)$ are determined in advance of path planning and task allocation. These positions are calculated using the belief map to distribute the low-level UAVs $\mathcal{U}_l$ over $\mathcal{S}$ in order to reduce the time for them to attend to tasks in future. Explicitly, we choose $L_{start}(u) \in \mathcal{L}_{start}$ for all UAVs such that the distance between the elements in $\mathcal{L}_{start}$ and the locations of the incidents $\{L(i_1), \ldots, L(i_n)\}$ is minimised. In the figure, these are labelled as $\mathbf{s_1}$ and $\mathbf{s_2}$. [Step 2]

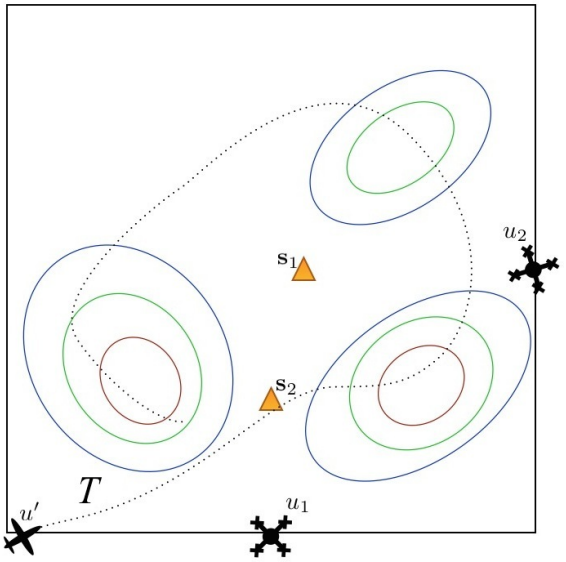
2. **Path Planning:** The exploratory high-level UAV u' plans and follows a path T calculated using the belief map $\mathcal{M}$ in order to maximise the probability of incident detection. [Step 2 onwards]

3. **Task Allocation:** UAVs are allocated tasks according to the set of current tasks $\boldsymbol{\tau} = \{\tau_{\mathbf{l}_1}^{u_1}, ..., \tau_{\mathbf{l}_m}^{u_m}\}$, where we use explicit notation to show that each task maps a UAV to any spatial location in $\mathcal{S}$ rather than directly to the location of an incident $L(i)$. Thus, a task $\tau_{\mathbf{l}}^{u}$ allocates a UAV $u \in \mathcal{U}_l$ to a position $\mathbf{l} \in \mathcal{S}$. These tasks are used to move the members of $\mathcal{U}_l$ to the prior placement positions calculated in Step 1 (see a detailed description in Section 3.2.1), then to move them to the locations of incidents once discovery has taken place. These chosen allocations are labelled $\boldsymbol{\tau}^*$. [Steps 3 and 4]

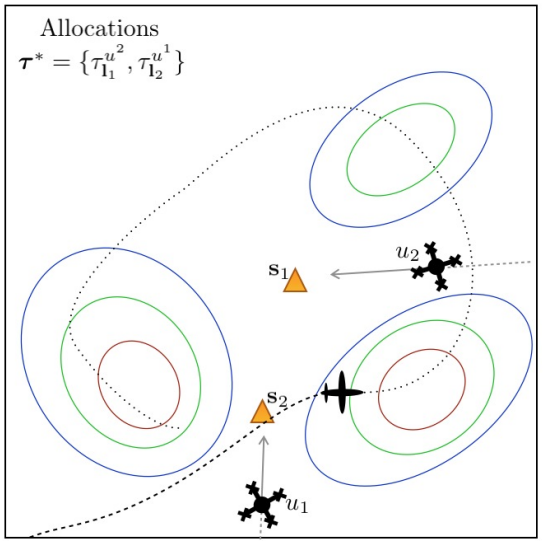
These stages are outlined in detail in the following three sections.



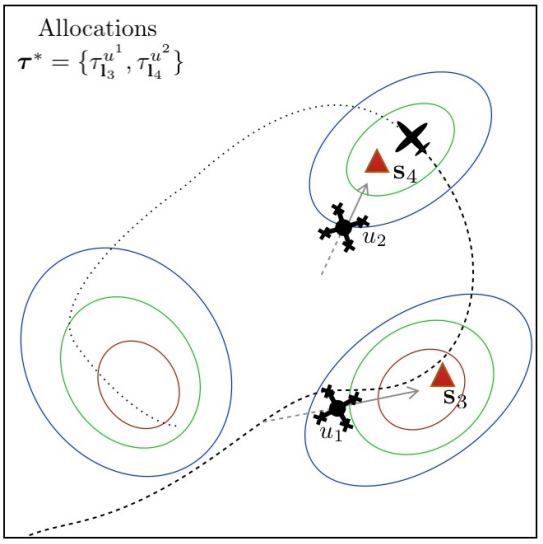
Step 1 - Belief data map $\mathcal{M}$ created over $\mathcal{S}$. Location of incidents i_1 and i_2 not known by UAVs.



Step 2 - Explorative path T planned, predictive placement calculated at $\mathbf{l}_1$ and $\mathbf{l}_2$ (shown as tasks s_1 and s_2).



Step 3 - Exploration begins. Predictive placement tasks allocated to u_1 and u_2 according to $\boldsymbol{\tau}^*$.



Step 4 - Incidents detected by exploratory UAV. New tasks s_3 and s_4 created and allocated. UAVs move to incident locations.

KEY:

	Contours indicating belief of incident in region. Belief indicated by blue (lowest) through green, to red (highest).
	Explorer (or high-level) UAV
	Low-level UAV
	Prior placement task locations
	Incident detection task locations

FIGURE 3.1: Outline of the processes and stages of the system

3.2.1 The Prior Placement Algorithm

We chose a simulated-annealing framework for the prior placement algorithm since it reliably produces near-optimal results with low computational overheads (as motivated in Section 2.2). The values obtained from $\mathcal{M}$ were treated as attractive components of the global potential function (driving the positions towards areas of high incident probability). Furthermore, the potential contains a component expressing a mutual repulsion between the low-level UAVs in $\mathcal{U}_l$, ensuring spacing between them. At this stage, we denote the set of positions of the low-level UAVs as $\mathcal{L} = \{L(u_1), \ldots, L(u_m) \mid u \in \mathcal{U}_l\}$. Consequently, we introduce our form of the potential function $V : \mathcal{S} \times \mathcal{M} \times \mathcal{L} \to \mathbb{R}$ as follows, where we refer to elements of the UAV position set $\mathcal{L}$ by their x and y coordinates: $(x, y) \in \mathcal{S}$:

$$V(\mathcal{M}, \mathcal{L}, x, y) \doteq \mathcal{M}(x, y) - \sum_{(x_\kappa, y_\kappa) \in \mathcal{L} \setminus (x,y)} \xi \left\| L(x, y) - L(x_\kappa, y_\kappa) \right\|^{-1} \tag{3.5}$$

Here, ξ is a constant used for tuning the repulsion heuristic of the algorithm and the sum is performed over all other UAV positions in the set $\mathcal{L}$ to ensure complete mutual repulsion.

Algorithm 7 details the simulated annealing method for placing UAVs given an (arbitrary) initial position set $\mathcal{L}_{init}$ (Line 2).[8] Descriptively, the algorithm proceeds as follows. At each step of the procedure, for each position in $\mathcal{L}$, a new pair of coordinates is selected at a fixed distance from the original position (Line 6) and their potential value is calculated according to V (Line 7). This calculated value is compared to the value of the potential at the original position (Line 10). If the new position results in an increased potential value, the new position is accepted. If it is not an improvement, it is accepted with a probability $e^{-\Delta V/\theta}$ (Line 16), where ΔV represents the change in potential value and $\theta \in \mathbb{R}^+$ is a temperature parameter. At each step, the temperature parameter is lowered according to a cooling schedule $\theta_i = \theta_{i-1} - \frac{\theta_{start}}{repeats}$ (Line 18), meaning that as the algorithm continues, position changes that lower the overall potential value become less likely. The cooling schedule, θ_{start} value, number of repeats, and step size are all heuristics in the algorithm, which we tuned to balance convergence time and optimality.

Once the positions $\mathcal{L}'$ are calculated, UAVs $u \in \mathcal{U}_l$ are allocated to them using the task allocation algorithm detailed in (Section 3.2.3). Specifically, a set of tasks is created and allocated $\boldsymbol{\tau} = \{\tau_{\mathbf{l}_1}^{u_1}, ..., \tau_{\mathbf{l}_m}^{u_m}\}$ exactly according to the max-sum algorithm (introduced in Section 2.1.3) used to allocate UAVs to incidents. This is detailed further in the task allocation Section (3.2.3) below.

During this placement of the low-level UAVs, the high-level explorer u' begins its path planning and following, which we now describe.

[8] Note that since the final positions calculated are close to optimal (i.e. providing short cumulative travel times to the locations of the tasks), they are in principle independent of the starting position set (an intrinsic property of simulated annealing over sufficiently long cooling schedules (Goffe et al., 1992)).

Algorithm 7 Predictive placement SA algorithm

//Map, list of UAVs, and schedule heuristics form the input arguments//
PlaceUavs($\mathcal{M}$, $\mathcal{L}_{init}$, $repeats$=5000, $stepsize$=0.5, θ_{start}=0.3)

1. $\theta \leftarrow \theta_{start}$
2. $\mathcal{L}' \leftarrow \mathcal{L}_{init}$
3. **for** κ **in** $\{1, ..., repeats\}$
4. **for** x,y **in** $\mathcal{L}_{init}$
5. //New position of one UAV calculated a fixed distance from previous position//
6. x', $y' \leftarrow$ `NewRandomPosition`($Distanceaway = stepsize$)
7. $origvalue \leftarrow \mathrm{V}(\mathcal{M}, \mathcal{L}', x, y)$
8. $newvalue \leftarrow \mathrm{V}(\mathcal{M}, \mathcal{L}', x', y')$
9. //Change accepted if new potential value is greater//
10. **if** $newvalue > origvalue$
11. Swap(x,$y \leftarrow x'$, y' **in** $\mathcal{L}'$)
12. **else**
13. //Otherwise, accepted with probability $e^{-\Delta V/\theta}$//
14. $\Delta V \leftarrow origvalue - newvalue$
15. $probability \leftarrow$ `exp`$(-\Delta V/\theta)$
16. **if** $probability >$ RandomFloatBetween$(0, 1)$
17. Swap(x,$y \leftarrow x'$, y' **in** $\mathcal{L}'$)
18. $\theta \leftarrow \theta - \frac{\theta_{start}}{repeats}$
19. **Return** $\mathcal{L}'$

3.2.2 The Path Planning Algorithm

To discover the locations of incidents I, the high-level UAV must explore the disaster space using the belief map $\mathcal{M}$, in order to maximise the probability of incident detection. In other words, the path which it follows must maximise the value of belief integrated along its length. Furthermore, cycles—that is, repeated loops in the chosen path—in one area of the space do not help in finding static incidents; so a balance must be created between exploring areas of high probability and not repeatedly moving over the same area of the space. As a result, we seek to create an objective function for use by our path planner which takes both of these factors into account, as well as move from the entropy map formulation used in previous literature to a belief-map regime.

To this end, we first consider that the result of the function $\mathcal{M}$ acting on a location in $\mathcal{S}$ is equivalent to a probability distribution (with higher values reflecting high probability of an incident– see Equation 3.2). Furthermore, we also consider that the goal of the planned path is to allow u' to explore areas of high incident belief (i.e. to position itself near to where incidents are most likely to be). Consequently, the values obtained from $\mathcal{M}$ can be used as a component of the objective function. Specifically, the high-level exploration seeks to maximise the value of $\mathcal{M}$ along its path, since this represents increasing the probability of locating an incident. Concurrently, we introduce a penalty for the creation of waypoints lying close to existing waypoints to avoid cyclic behaviour by driving the planner away from areas that have been explored already. We define this penalty function C for a given waypoint $\mathbf{s}'_\kappa$ in the form of symmetrical Gaussian functions[9] around each previous waypoint in the path, which are summed to produce a negative output from the function $C : T \to \mathbb{R}^-$:

$$C(\mathbf{s}'_\kappa \mid T) \doteq -\sum_{prev=1}^{\kappa-1} \omega_{prev} \mathcal{N}(\mathbf{s}'_{\mathbf{prev}}, \sigma_C^2) \tag{3.6}$$

Here, we denote the normal Gaussian distribution as $\mathcal{N}$, with the mean argument as the location of the previous waypoint $\mathbf{s}'_{\mathbf{prev}}$, and a fixed covariance argument σ_C^2 (for symmetry around each point). In more detail, the weight ω_{prev} is calculated from the distance ∂_{prev-i} of each previous waypoint in the path to $\mathbf{s}'_\kappa$, and a preset scale factor ϱ (used to adjust the magnitude of the repulsive component) shown in Equation 3.7. The weight of the function is truncated in order to streamline computation time[10] to positions a distance ζ from $\mathbf{s}'_\kappa$.

[9] We select Gaussian functions to show that the cost incurred from each existing waypoint diminishes with distance.

[10] A more rigorous approach would accurately reflect the expected occurrence of an incident in an area, according to some prior distribution over future events—however this is out of the scope of our work.

$$\omega_{prev} \doteq \begin{cases} \frac{\varrho}{\zeta}\partial_{prev-i} - \mathbf{s} & \text{if } \partial_{prev-i} < \zeta \\ 0 & \text{otherwise} \end{cases} \tag{3.7}$$

Ideally, one would integrate the objective function along the length of the path to calculate its overall contribution since this represents the infinitesimal sum of contributions of each point on the belief map. However, we take this opportunity to introduce some simplifying heuristics to better suit the RRT algorithm to our particular scenario. Specifically, since the path is calculated in discrete waypoints a short distance apart, we need not fully compute the integral value. Instead, we can approximate it (and thus substantially reduce computational complexity) by summing along the waypoints in the path. This provides an accurate estimate of the integral as long as the scale of the features of the distribution are sufficiently larger than the waypoint separation, or, in other words, as long as there are no sharp increases or decreases in the function's value in the path between waypoints. This approach also simplifies point-wise calculation of new waypoints as their contribution to the objective function can be simply added to the current value, rather than needing to integrate. The objective function at a point in $\mathcal{S}$, is then simply the maximum of the sum of the value obtained from $\mathcal{M}$, and the cost value from C, and is defined in the algorithm description as the maximum of the utility function U.

$$\max_{\mathbf{s}'_\kappa} U(\mathbf{s}'_\kappa) \doteq \max_{\mathbf{s}'_\kappa} \left(\mathcal{M}(\mathbf{s}'_\kappa) + C(\mathbf{s}'_\kappa \mid T)\right) \tag{3.8}$$

We chose an RRT framework based on the modified growth mechanism of Scerri et al. (2008) (shown in detail in Section 2.3.3) for path planning. As discussed above, RRT is suitable because it accounts for the lack of a definite end-goal position in the space, and can grow over the continuous belief map we use here. The exact implementation of the mechanism is different from that outlined in the original paper (Algorithm 3) only in the use of a numerical utility (U) to be maximised instead of a cost to be minimised. In other words, we replace Line 23 with: **if** $n'.utility > best.utility$, and references to "cost" with "utility". This is logical in the context of a belief map over which we wish to maximise coverage of areas of high-belief of tasks, and does not impact the form of tree growth. Note that we refer to waypoints as *nodes* in the algorithm, in keeping with the notation used by Scerri et al.

Finally, we note (as before) that we assume perfect incident detection within a fixed range of the exploratory UAV, as described in Equation 3.1. Once an incident is deemed detected, task allocation of the low level UAVs is performed according to the algorithm shown in Section 3.2.3 below.

3.2.3 The Task Allocation Algorithm

The task allocation algorithm we chose for distributing discovered incidents among UAVs is the same as outlined in Section 2.1.3; namely a discrete max-sum algorithm. This was chosen for its decentralised form, empirically established performance guarantees, and robustness to addition of new tasks. To implement max-sum, we need to specify the functions computed at the function nodes, to be sent as messages to the variable nodes. To this end, we select a function to cause the algorithm to reduce the distance the UAVs travel—and thus the duration—before arriving at an incident, in accordance with the goal of task assignment in Equation 3.4. Since the function will be maximised, we formulate this as a reciprocal distance relationship between UAV position and task position (i.e. maximising the function will minimise the UAV travel time), truncated to a maximum heuristic value γ chosen to avoid unbounded results as the UAV approaches the task:

$$f_j \doteq \begin{cases} \gamma & \text{if } \|L(i) - L(u)\|^{-1} > \gamma \\ \|L(i) - L(u)\|^{-1} & \text{otherwise} \end{cases} \tag{3.9}$$

This function then forms the components of the global utility function $F = \sum_j f_j$ originally described in Equation 2.1, and the max-sum messages exchanged are exactly as described in Section 2.1.3, outputting an optimal allocation $\boldsymbol{\tau}^*$ which maximises F. This results in the maximisation of the total truncated-reciprocal distance[11] between the incidents and UAVs, giving us our desired behaviour.

Additionally, the prior placement algorithm outlined in Section 3.2.1, uses this task allocation framework once positions for the UAVs are calculated, to distribute them among the chosen locations. We achieved this by creating tasks at the predicted locations for the UAVs to move towards, which were then removed once the UAVs reached their designated location. The re-use of the same max-sum framework in both areas of the mechanism ensures the placement of UAVs benefits from the same optimality and speed as the incident task-allocation (as well as reducing the need for further algorithms).

3.3 Experimental Design

We conduct three experiments in simulation to test and benchmark the algorithms outlined in the previous sections:

1. We first examine the effectiveness of our RRT planner in searching for incidents given a belief map. To do this, we compared RRT to a simple lawnmower coverage

[11] We reiterate here, that the truncation avoids unbounded results as the UAV approaches the location of the task, which would cause $\lim_{L(u)\to L(i)} \left(\|L(i) - L(u)\|^{-1}\right) \to \infty$.

process (discussed originally in Section 2.3.1) and a situation with perfect information where the UAV moves immediately to the locations of the incidents, $L(i)$. We do this to show that the RRT planner is suited for use over a belief map, since originally it was used in the slightly different entropy-map scenario (discussed in Section 2.3.3), and that it outperforms the simple coverage scenario.

2. We next examine the behaviour of the path planner further, by testing how the time to respond to incidents is affected by the form of the belief map (i.e. the covariances of the Gaussian components). We do this to examine the relationship between the form of the belief data, and the effect on the efficiency of the planning algorithm; thus reflecting the variations in real-world belief map scenarios the algorithm may be applied to. Specifically, we examine how the spread of the distribution around isolated incidents affects the speed at which the high-level UAV can locate them.

3. We integrate all the components of our combined system and compare it to several variations that replace the max-sum allocation with a greedy mechanism, and the RRT planner with a coverage method; as well as removing the prior-placement algorithm. We do this to examine whether our combined method is more effective at minimising t than these benchmarks, and also to examine the individual contributions to the performance of the system by each component.

Since the aim of our work is to reduce the response time t, we used a built-in software based timer to record the duration of completion of the three experiments. The experiments themselves are discussed in details in Section 3.4, where we also elaborate on the benchmarks used.

For the purpose of these experiments, we model the belief map $\mathcal{M}$ as a sum of bivariate Gaussian distributions in 2D Euclidean coordinates, since probability distributions over a space commonly resemble the normal—or Gaussian—form (Caselli et al., 2001). Incident locations were selected by sampling from the distribution. In Equation 3.10 below, we denote the relative weight of the ith component of the distribution as λ_i, and the component's mean vector and covariance matrix as μ_i and σ_i^2 respectively.

$$\mathcal{M} \doteq \sum_i \lambda_i \cdot \mathcal{N}(\mu_i, \sigma_i^2) \tag{3.10}$$

For the message passing between path planner and path execution process, and between agents for the task allocation algorithm, the open source ROS[12] package was used with Python[13] scripts. These were chosen for both their versatility, and also their resemblance to APIs deployed on real UAV platforms. Each agent operated its own separate computational process to best reflect their physical separation in real world situations,

[12] `http://www.ros.org/wiki/`.
[13] `http://www.python.org`.

and communications are simulated via ROS message threads. Exact specifications of repeat runs are detailed below. We measure path planner performance based on time to discover all incidents, and the performance of the whole system by how long it takes for all incidents to be attended to.

3.4 Results and Empirical Evaluation

Below, we describe in detail the results from the three experiments we performed; namely two specific examinations of the performance of the path planner and a combined test showing the efficacy of the whole mechanism.

3.4.1 Path Planner Performance

Our first experiment tested the performance of incident discovery—ensuring the viability of our belief-map using RRT implementation—via the methods of:

- RRT (presented in Section 3.2.2).
- A lawnmower coverage method, moving in stripes to cover the whole space (Section 2.3.1) shown schematically in Figure 3.2. This was uncoordinated, and began from a randomised starting location.
- A perfect information straight-line approach that directed u' directly to the incidents.

Results were taken over a two-component Gaussian belief map which was kept constant throughout the experiment. This map was chosen primarily for computational simplicity in establishing initial performance measures, while not being as straightforward for planning as a single Gaussian distribution. We collected results independently from 200 runs for statistical significance.

In more detail, we benchmarked our algorithm against an algorithm with perfect information of incident locations, where the path simply traverses the exact incident positions via straight lines, and a naïve lawnmower-style coverage method. This latter method runs in parallel paths up and down the y axis, while moving a small distance in the positive direction on the x axis when it reaches the top and bottom boundaries of the space. The distance between these paths (namely, the distance moved along the x axis after each vertical section) is equal to the sensing area width δ of the high-level UAV, ensuring complete coverage.

Results are presented in Figure 3.3, along with error bars showing the standard errors of the means of the results. To interpret these results we note that as more targets

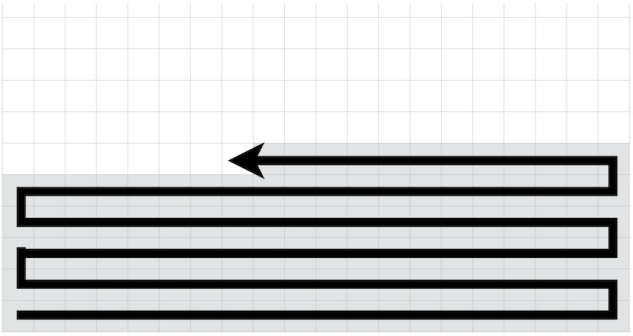


FIGURE 3.2: Illustration of the form of a typical "lawnmower" style sweep search, with incremental movement across the search space.

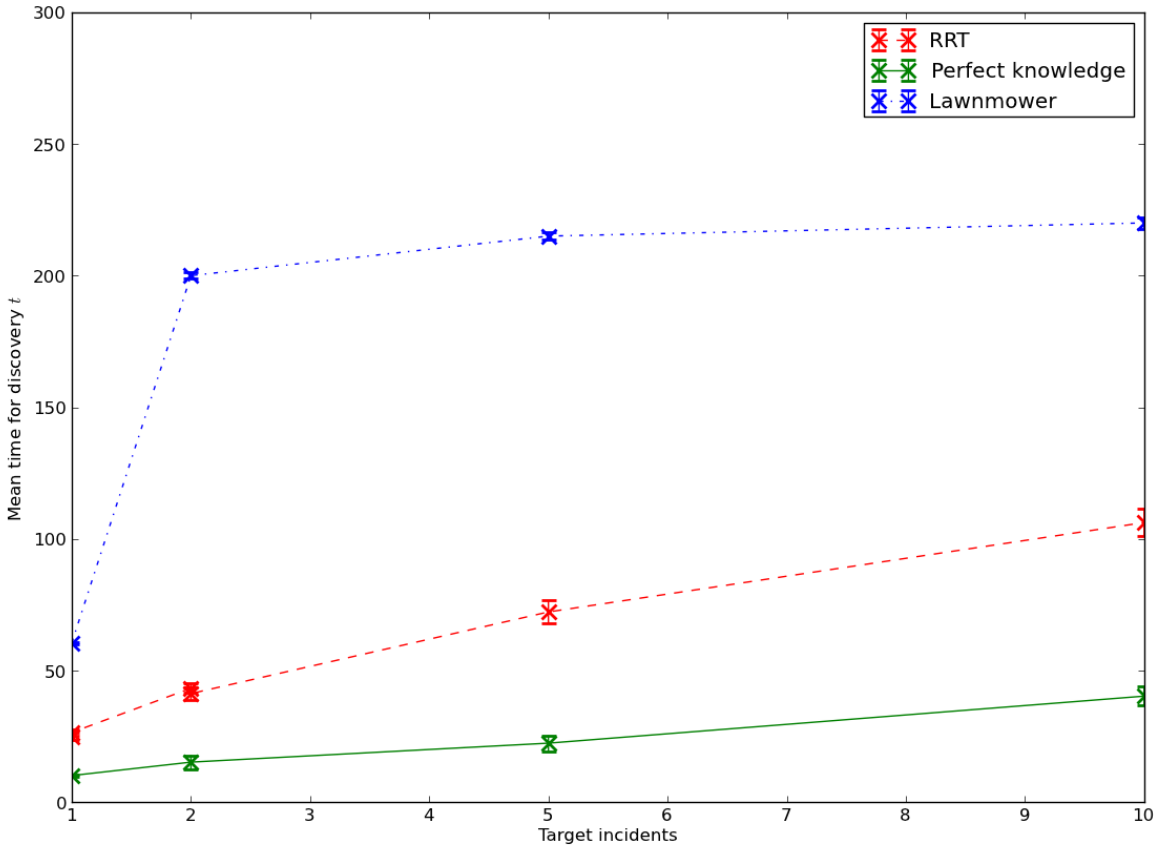


FIGURE 3.3: Results for discovery time of 1, 2, 5, and 10 incidents using either the RRT method, perfect information, or a simple lawnmower procedure. Error bars calculated using standard error of the mean.

are introduced, the relative density of targets in the space increases, resulting in relatively less sharp increases in exploration time (since the explorer need not necessarily move further to discover more tasks). This is particularly apparent in the lawnmower case. Since the lawnmower traverses the space at a continuous rate, the time it takes to reach incidents depends entirely on their distance from its starting position. The RRT algorithm substantially outperforms this method by targeting only those areas of high

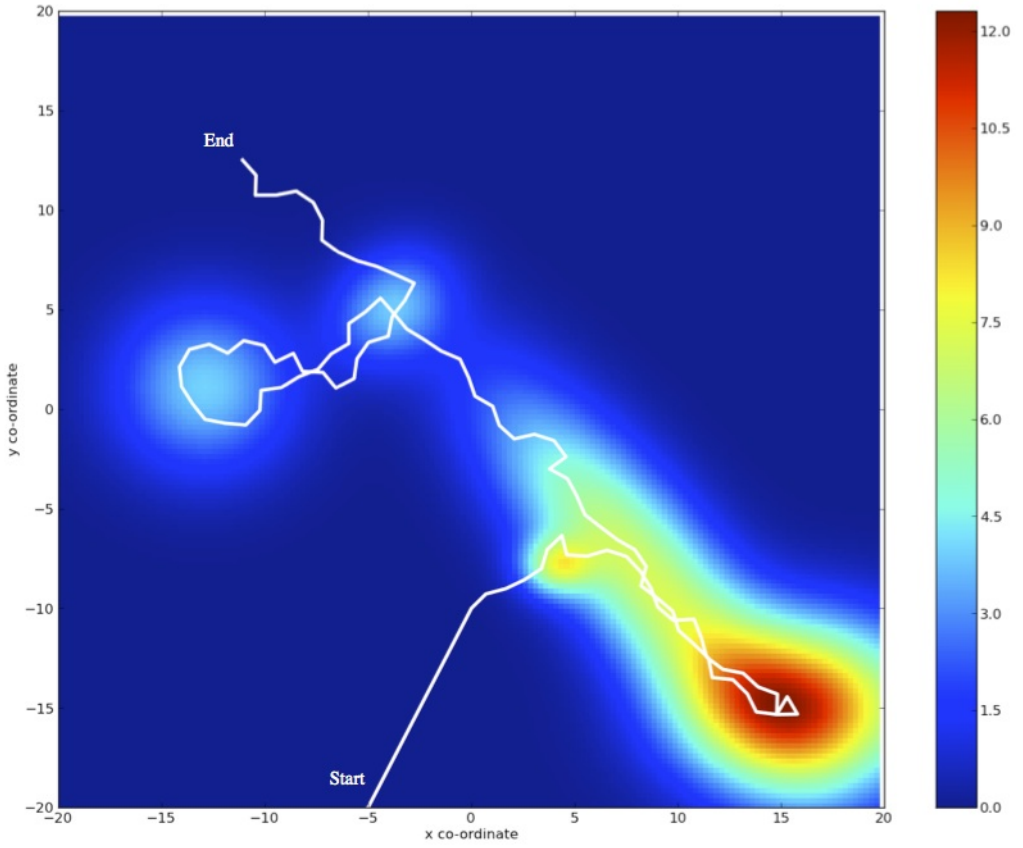


FIGURE 3.4: Example belief map with exploratory path overlay, showing the exploratory planner's ability to seek out the various areas of higher belief in the Gaussian mixture. Colour scale represents probability density.

belief, performing more than twice as well as the lawnmower technique for the ten target case (see Figure 3.3).

For illustration purposes, an example of an exploratory path produced over a belief map is shown in Figure 3.4, where areas of high belief are shown in redder shading. Notice that the path tends to follow the peaks in the distribution and has reached the two smaller Gaussian components in the left of the search space.

3.4.2 RRT Performance Dependence on Belief Map Spread

In order to determine the effect of the size of the covariances of the distribution on incident discovery time, an additional experiment was performed on single incident discovery. In reality, a wider spread in a distribution could correspond to less precise knowledge of the belief of an incident. We collected data over 600 runs averaged over four different distribution widths[14] at increasing distance from the origin of the explorer. From the results in Figure 3.5, we can see that narrower Gaussian distributions (i.e. with smaller covariances) increases discovery time; this is because wider distributions allow the RRT

[14]i.e. increasing values of the variance of the Gaussian component in $\mathcal{M}$ (Equation3.10).

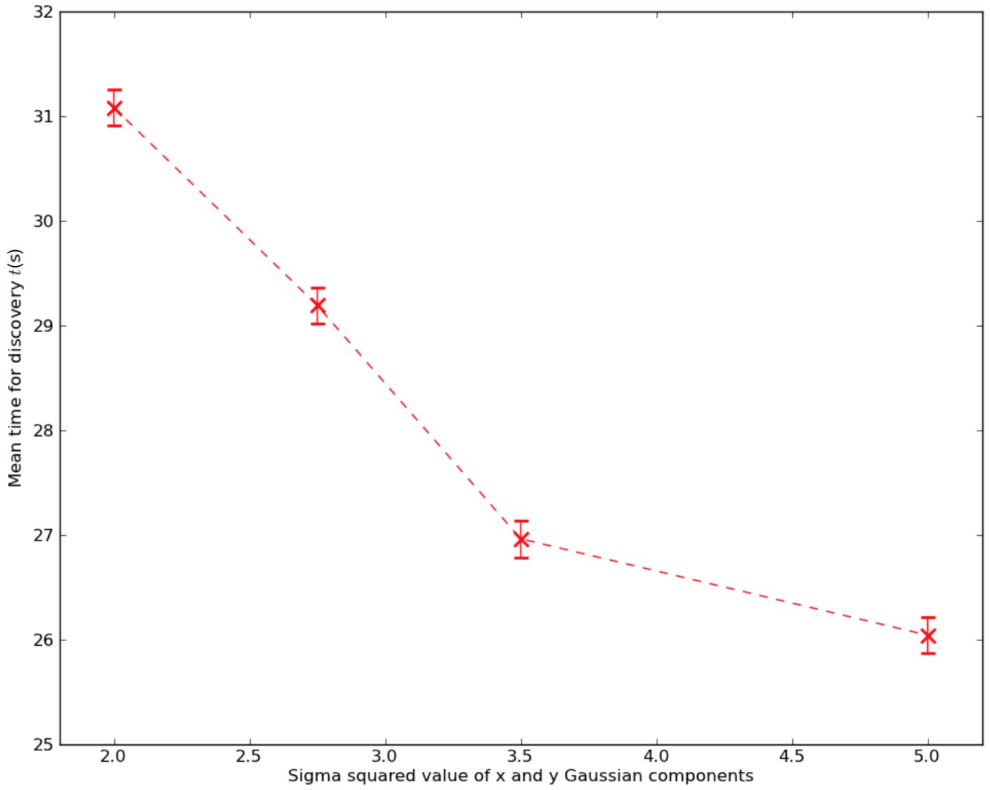


FIGURE 3.5: Discovery time differences due to change in width of Gaussian component. Errors bars calculated using standard error of the mean.

planner to obtain rewards from moving in the direction of the incident at greater distances. We refine our path planning models in subsequent chapters, in part to address this incongruity: one would expect perhaps, that in a real scenario a wider distribution indicated greater uncertainty, and would therefore lead to longer discovery times.

We discovered from empirical evaluation of the trees being produced by the RRT algorithm (that produce the path T) that minimal branching occurred compared to that described by Scerri et al. (2008) in their original work. Specifically, branching paths in the tree were seldom longer than a single waypoint in length. This indicates the tree is growing over a restricted region of $\mathcal{S}$ and therefore possibly failing to reach isolated sharp peaks. We use this knowledge to improve on the exploration reformulation in our extended, coordinated, exploration algorithm in Section 4.2.

3.4.3 Combined Mechanism Performance

Here, we evaluate the performance of all three components outlined in Section 3.2, to determine the benefits of our combined scheme using the path planning, task allocation, and prior placement techniques to reduce the travel time t of UAVs in $u \in \mathcal{U}_l$ to position themselves at the locations of the incidents $i \in I$. In this way we show the advantage of our system and its applicability to real world disasters requiring fast response to

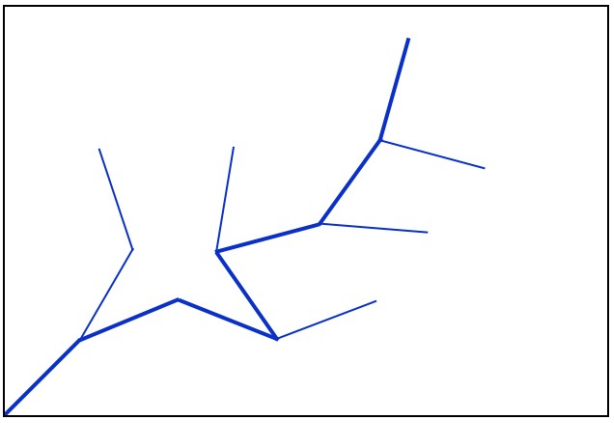


FIGURE 3.6: Sample of branching RRT path plan, with thick line showing selected trajectory. Note the relatively small number of branches.

incidents. Specifically, six methods were compared to determine which produced the lowest total time for incident discovery and attendance. The results are shown in Figure 3.7 for the following scenarios:

	Explorative algorithm		**Predictive**	**Task allocation algorithm**	
Abbreviation	Lawnmower	RRT	**placement**	Greedy	Max-sum
PI	*Perfect information scenario*				
LG	✓			✓	
RG		✓		✓	
RM		✓			✓
RPG		✓	✓	✓	
RPM		✓	✓		✓

TABLE 3.1: Table of abbreviations in the description of the experiment in this section, and how they relate to the algorithms used.

Regarding the benchmarks used, the lawnmower method was identical to that employed in Experiment 1, whereas the task allocation benchmarks were a simple greedy mechanism, and the result of the perfect-information scenario described below. The greedy mechanism relies on each low-level UAV seeking to minimise its own f_j value (Equation 3.9) without regard for the other UAVs in the area (i.e. without any co-operative message passing). The absolute lower time bound for incident attendance—defined in our scenario as perfect-information (PI)—by the prior placement of UAVs over the incidents before exploration has concluded: namely a prior placement allocation coinciding exactly with the incidents themselves. Additionally, as above, the exploratory UAV moves in straight lines from the exact location of incident-to-incident. In contrast, those combinations without any prior placement used randomised starting locations for the task responder UAVs. We note the lack of comparable state-of-the-art work in this area since such a combination of work has never been attempted previously (as discussed in Section 2.4).

The experiment was performed over a constant number of five incidents, with a set of five task-allocation UAVs, so that we did not have to consider the length of time a UAV

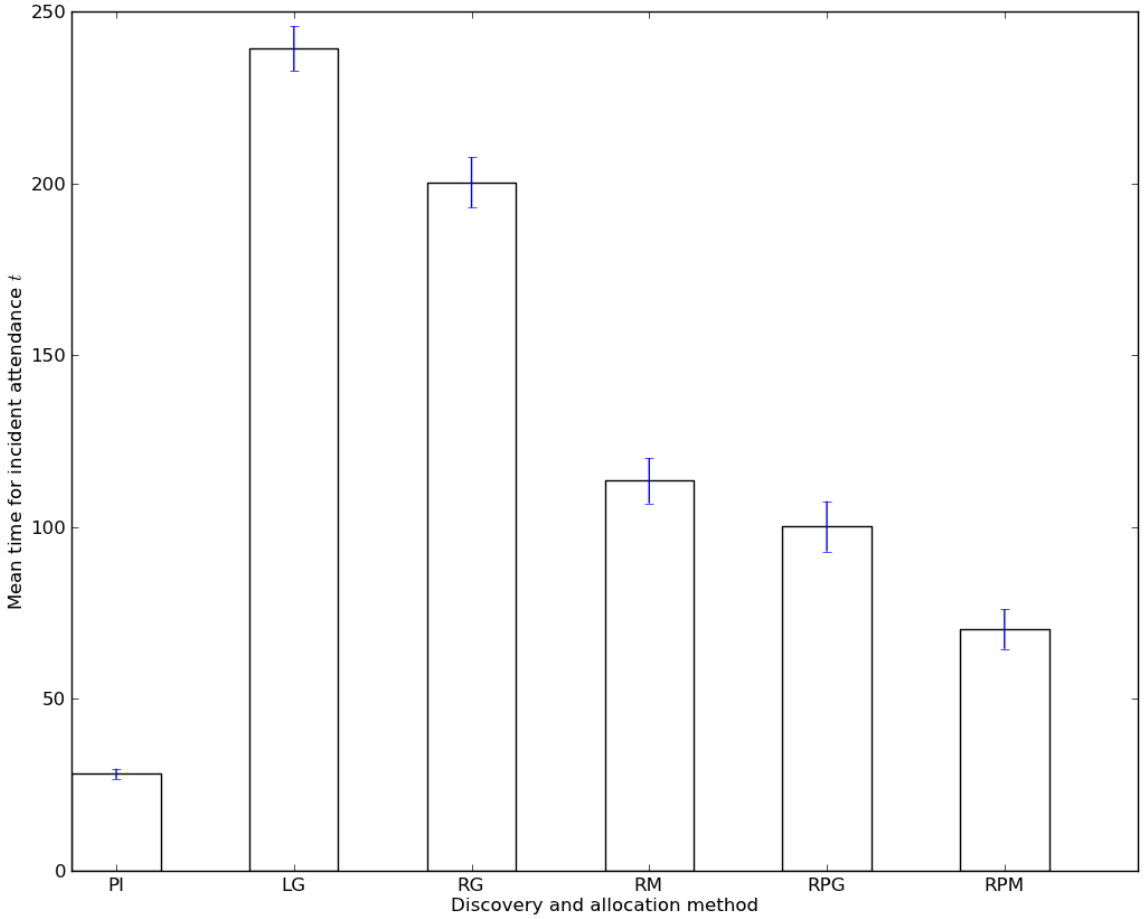


FIGURE 3.7: Combined time for different mechanisms to discover all incidents and have them attended by UAVs. Times in seconds.

was positioned over an incident to complete its task (as discussed in 3.1). Performance was measured against cumulative incident response time as described in Equation 3.3, with lower values being more favourable.

Experimental results showed an improvement in lower values of t over simpler alternatives to the framework we propose. As expected, the complete RPM procedure outperforms all other combinations of process considered here except that with perfect information. The improvement varied from a factor of (at most) 350% between RPM and LG, and at least 25% between RPM and RPG. Even the RG method (which discounts predictive placement and uses only greedy task allocation) outperforms the LG method that does not use the explorative RRT path planner.

The slightly larger error bars for the RRT methods reflects the stochastic nature of this method. The addition of max-sum task allocation or predictive placement to the RRT algorithm resulted in a time reduction of almost 50% in each case (compare RG to RM and RPG), clearly demonstrating the benefit of employing a non-greedy task allocation method and a prior placement algorithm. Although the prior placement algorithm appears to be more valuable overall to the mechanism (with RPG outperforming RM by

an average of 13.3 seconds) the two results have overlapping error bars, leading us to conclude that they are of almost equal importance in the overall mechanism. The RPM mechanism represents a further 30% increase in performance over the next best (RPG) mechanism, suggesting that the contributions from the prior placement and the max-sum task allocation are not linearly additive because of diminishing returns from the combination of multiple mechanisms.

Finally, we note that the perfect information procedure performed on average 42.1 seconds faster than our RPG mechanism; suggesting an upper-bound to the solution of our problem. We conclude that the combined mechanism we present is shown to perform in accordance with Criteria 1–2 (see Section 3.5 for further discussion).

During the course of these experiments, we observed that an emergent feature of the RRT paths planned was the relatively small amount of branching in the tree created before a path was selected (as discussed previously in Figure 3.6), regardless of the tuning of the various parameters such as the priority calculation function and the previous waypoint cost weighting parameter ζ. Although the path planning performed well, it was seen that branches were rarely longer than a single waypoint away from the main path. Branching behaviour is desirable since generally, many different paths must be considered to find the optimal route through the space and a lack of branching severely restricts the overall area covered by the RRT process. This was due to the interplay between the simple heuristic for driving new paths away from previous paths (outlined in Equation 3.6) and the utility calculation. We address this issue explicitly in our reformulation in Section 5.3 where a more principled update mechanism for areas of the map that have been previously explored solves this anomalous behaviour and results in further branching into the space. Another undesirable behaviour was the occasional instance of the path planner getting stuck in a certain position and repeatedly expanding a single node despite the priority penalties for doing so. This is again related to the calculation of the cost function (Equation 3.6) of creating a new waypoint in T close to existing points in the path, since previous nodes would necessarily have more neighbours than those at the frontier of the path. In subsequent work (beginning with Section 4.2), the refinement of our path-planning algorithm addresses this problem implicitly by reformulating the algorithm in line with the extension of our scenario to multiple explorative agents.

3.5 Summary

In this chapter, we have presented the method and results for the first combined mechanism for the discovery and allocation of tasks for a team of heterogeneous UAVs, and demonstrated its usefulness in simulation. In particular, we have shown that the system outperforms lawnmower path and greedy task allocation algorithms, as well as providing a further time saving in incident response through the use of a prior placement algorithm.

Referring to our requirements detailed in Section 1.1, we have demonstrated the following in terms of the research criteria:

Criterion 1: Heterogeneity Throughout each of our experiments we have consistently treated our simulated UAVs as heterogeneous between the high-level $u' \in \mathcal{U}_h$ and low-level $u \in \mathcal{U}_l$ vehicles, by assigning them different algorithms to determine their behaviour (i.e. RRT for the high-level, and prior-placement and max-sum for the low-level).

Criterion 2: Belief Data Use Belief data has been used in our assignment of prior placement positions to the low-level UAVs, and in planning the explorative path of the high-level UAV (see Lines 7 and 8 in Algorithm 7 and Equation 3.8 respectively). We fulfilled the need for responding to tasks by assigning incidents discovered by the high-level UAV as tasks to the low-level UAVs in simulation. These were then attended while trying to minimise the total time for the group according to Equation 3.3. We fulfilled the need for UAV prior-placement by using a simulated-annealing algorithm in our simulations to generate positions using the belief map, and the same max-sum task allocation framework for incidents to drive the low-level UAVs to these locations.

Furthermore, we have fulfilled the research requirements detailed in Section 1.1 to the following standards:

R1: Decentralised Autonomy We have shown decentralised autonomous behaviour in the low-level UAVs by using a fully decentralised task allocation algorithm to coordinate multiple heterogeneous simulated UAVs. We discuss the inclusion of further high-level UAVs to aid exploration in Chapters 4 and 5 below in a discretised and then a continuous search-space respectively.

R2: Coordination UAVs in our scenario coordinate explicitly to perform task-response to incidents as they are discovered, with the global goal of minimising the time taken for each incident to be reached. Nonetheless, there remains the need for coordinating explorative agents as per this requirement, and Criterion 3.

R3: Scalability We have shown the capability of the system to cope with seven UAVs in total, and detail in subsequent chapters how we extend the algorithm to allow multiple explorative UAVs in addition to those completing tasks in this model.

Additionally, to inform future work on improving our exploration algorithm, we have noted emergent behaviour of small amounts of branching in the RRT planner. In order to account for this, and in order to meet Criterion 3, we go on to reformulate our explorative algorithm to allow for coordination in Section 4.2.

Having established the groundwork for the combined mechanism, we next seek to address the situation where there are multiple explorative UAVs present. This is a logical progression towards a more realistic system since we already account for multiple task-attending UAVs, and the combination of algorithms should be robust to the different availability of differing numbers of UAVs in a disaster space. Multiple explorative agents require a substantial reformulation of the exploration problem to overcome situations where multiple agents tend to explore the same area. Furthermore, we have not yet considered in detail the problem of incident discovery, since we currently use a simple distance metric to determine whether an incident has been identified by the exploratory UAV—i.e. an incident is considered discovered when the exploratory UAV is within a certain range of its location (representing the sensing range). Considering real-world applications, the exact nature of an 'incident'—specifically those at a single fixed location—in a disaster scenario is not intuitive and can be regarded as a relatively crude representation of the complexities of a broad area with various likelihoods for casualties or victim deaths.

In the next chapter, we seek to address both these issues by constructing a more nuanced model of the dynamics of the disaster environment, formulating the problem of exploring this area for the purposes of saving as many lives as possible, and introduce our solution to the problem in the form of a multi-UAV explorative path-planner that builds on work introduced in Section 2.3.

Chapter 4

The Coordinated Explorative Path Planning Mechanism

So far, we have assumed a single explorative UAV to be responsible for locating incidents and initiating them as tasks for the lower-level UAVs to attend. In order to discover incidents more quickly, multiple coordinated explorative UAVs may be deployed in a disaster situation (as per Criterion 3 laid out in Section 1.1). This would allow quicker exploration of an area and thus reduce the total time to discover incidents. However, if further explorative UAVs are introduced, an addition to the path planning is needed to establish the co-operative method by which the explorers traverse the environment. We seek to address this in the following sections where we introduce the first coordinated tree-search path planning algorithm. In order to do so, we note some restrictions to simplistic approaches that we must overcome.

In more detail, a naïve approach would involve a simple partitioning method where the area is divided between the explorers who then have no further interaction. However, in general, this approach would not be optimal since it neglects the benefits of co-operative targeting of the areas of the map with highest belief; namely that different UAVs can respond to disparate areas simultaneously. These areas may not be distributed evenly, and will require quick attendance by UAVs. We outline how our coordinated method solves these problems in Section 4.2 and evaluate its performance in Section 4.3.

Furthermore, through careful formulation of the problem and further consideration of the type of data available to responders, we introduce a more concrete motivating scenario; where we specifically consider the different data available to first responders from crowd reporting. As a result, we formulate a coordinated UAV explorative path planner that utilises a modified form of belief information to model realistic disaster scenarios. We build this onto an MDP framework according to our findings from subsection 2.3.4, because it allows explicit forward-planning and step-wise coordination.

4.1 Coordinated Path Planning Problem Formulation

The overarching aim of disaster response work is to minimise the loss to human life in the disaster area. To this end, we consider the exploration of a disaster situation where we require that UAVs focus on areas of perceived danger, but also where these regions intersect with likely occupation by people (Adler et al., 2014; Macintyre et al., 2011; Harvard Humanitarian Initiative, 2011). The rationale here, is that, data about a region containing known hazards (for example, high levels of radiation or severe building damage) is only useful in preserving human life when it is known or believed that there are likely to be persons in the vicinity of a hazard; or will be at some future time. Moreover, research shown that quick attendance to casualties in disasters significantly increases survivability (Fawcett and Oliveira, 2000; Hoffmann and Tomlin, 2010; Tadokoro, 2009).

We give the example of radiation as a possible manifestation of *danger* in a disaster scenario. In principle, we can extend this to any phenomenon that is present over an extended area and represents some risk to human life. This general approach could then represent several types of hazard in a disaster area, such as flooding, fire, chemical spills, or risk from earthquake-damaged buildings. At this point, we consider both the location of such hazards and the distribution of people as static. Such an assumption can be justified in scenarios with slow-changing conditions (relative to the time taken by the UAVs to explore) and where people are likely to be trapped in certain locations.[1] This work is the first example of danger and population data being used in this way. We also note the decomposition of the search space into a cellular form, to accommodate our MDP solution.

As a result, we consider the problem of exploring a uniform $\chi \times \psi$ sized grid world, $\mathcal{S}$ formed of cells,[2] $c_{xy} \in \mathcal{S}$. Each cell c_{xy} contains an *unknown* number of people, $p_{xy} \in \mathbb{Z}^*$, and a scalar value $d_{xy} \in [0, 1]$, denoting the *danger* in the cell as the probability of each person occupying that cell dying in the next time step (i.e. with a danger of $d = 0.1$ a cell initially containing 100 people would see 10 of them "dying" in the next time step). Time steps are denoted by an integer value $t \in \mathbb{Z}^*$, with subsequent steps referred to by adding integer values; for example $t + 1$. If a value depends on time, this is denoted in parenthesis; i.e. $p_{xy}(t)$.

The area $\mathcal{S}$ is explored by a UAV in the set of UAVs $\mathcal{U}_h = \{u_1, \ldots, u_\eta\}$ that traverse $\mathcal{S}$ from cell to cell once per time step.[3] Each UAV is equipped with sensors capable of accurately detecting people and danger inside one cell at any given time step: i.e.

[1] In particular we believe this specifically relates to disasters caused by—for instance—flooding or earthquakes, where building collapse or environmental hazards prevent survivors from being able to move around freely.

[2] Cells are indexed for their horizontal and vertical position respectively by xy.

[3] Since in this chapter we only consider the UAVs introduced in Section 3.1 as high-level explorers (a single example of which was previously denoted u'), we henceforth omit the apostrophe mark for simplicity and refer to a single UAV as u or u_κ, where κ denotes a generic index.

they return the values of p_{xy} and d_{xy} with no uncertainty (we discuss the implications of this below). Each UAV, u_κ, selects its own trajectory $T_\kappa \subseteq \mathcal{S}$ with contiguous borders, through the area, forming the set of all UAV trajectories $\mathbf{T} = \{T_1, \ldots, T_\eta\}$. To address the issue of double-counting the values in each cell,[4] we introduce the union of all the trajectories in the set $\mathbf{T}$ as $\mathcal{T}(\mathbf{T}) = \bigcup_{T_\kappa \in \mathbf{T}} T_\kappa$. Additionally, we impose constraints on the length of each UAV's planned trajectory to account for limits on battery life. If the maximum number of cells that can be traversed due to the battery limitations of the κth UAV is denoted $b_\kappa \in \mathbb{Z}^+$ then the maximum trajectory length is simply $\mid T_\kappa \mid \leq b_\kappa \forall \kappa \in \{1, \ldots, \eta\}$.

With this in mind, we formulate the multi-UAV exploration problem as a Markov Decision Process (MDP) (as per our discussion in Section 2.3.4), comprising a tuple $\langle S, A, R, P \rangle$ of states S, actions A, reward R, and transition probabilities P, which we define below. Since the UAVs are not aware in advance of the ground-truth values of p_{xy} and d_{xy} in each cell, computations are instead made using prior belief-data about the expected number of people $\bar{p}_{xy} \in \mathbb{R}^*$ and the expectation value of the probability of death (also known as the *death-rate*) in a given time step:

$$\bar{d}_{xy} \in [0, 1] \tag{4.1}$$

We also consider a binary variable $o_{xy} \in \{0, 1\}$ denoting the *visibility* of the cell: i.e. whether it has been observed by a UAV. This allows us to construct the tuple s_{xy} to denote the state of a cell c_{xy}:

$$s_{xy} = \langle \bar{p}_{xy}, \bar{d}_{xy}, o_{xy,} \rangle$$

The set of these for all cells in $\mathcal{S}$ forms the global state variable of cells $s = \{s_{00}, \ldots, s_{\chi-1\psi-1}\}$. With this constructed, we next formulate an update procedure for the expectation value of the number of people in a given cell after a given time step t by computing the product of the current expected number of people and the probability of survival (i.e. the complement of the probability of death):

$$\bar{p}_{xy}(t+1) = \bar{p}_{xy}(t)(1 - \bar{d}_{xy}) \tag{4.2}$$

Here, $\bar{d}_{xy}$ for the next time step is dependent on whether a cell has been observed. If so, we consider the danger to reduce to zero, since first responders can then be aware of the need to rescue the people occupying that cell:

[4] Specifically, we seek to avoid repeated observation of the same high-value cells by different UAVs. Thus, by taking the union of trajectories, we ensure only unique observations contribute to the utility function.

$$\bar{d}_{xy}(t+1) = \begin{cases} 0 & if\ o_{xy}(t) = 1 \\ \bar{d}_{xy}(t) & otherwise \end{cases} \tag{4.3}$$

This assumption need not hold in general, however it is a convenient way of indicating that no further utility can be derived from revisiting a cell once it has been observed. In order to correctly describe the likelihood of incidents occurring once an area has already been explored, significantly more data and examples from real disasters would need to be studied and modelled; something beyond the scope of our contributions.

We note for completeness the logical extension of equation 4.2 to times at an arbitrary point in the future t':

$$\bar{p}_{xy}\left(t'\right) = \bar{p}_{xy}(t)\left(1 - \bar{d}_{xy}\right)^{t'-t} \tag{4.4}$$

We record the set of positions of each UAV $u_\kappa \in \mathcal{U}_h$ at time t as $g_\kappa(t) \in \mathcal{S}$, which we denote as members of the vector of all UAV positions:

$$g(t) = (g_1(t), \ldots, g_\eta(t))$$

where the indices on each g correspond to the indices of the UAV at that location. Additionally, we record the set of unique UAV locations as the union of the elements of g as:

$$G(t) = \bigcup_{\kappa=1}^{\eta} \{g_\kappa(t)\}$$

Thus, the state of the map at a time t is a tuple $\tilde{s}(t) \in S$ comprising the state of each cell, and the position of each UAV:

$$\tilde{s}(t) = \langle s(t), g(t) \rangle$$

The action vectors enabling the UAVs to transition from cell to cell are defined as $a_\kappa = (\leftarrow, \rightarrow, \uparrow, \downarrow)$ for each UAV u_κ, (each arrow representing the direction of motion) with the constraint imposed that the available actions are restricted to agents at the edge of the grid world (i.e. explicitly where the UAV occupies a cell c_{xy} where $x = 0$ or χ, or $y = 0$ or ψ) so that they do not have the action available to cross out of the grid area. At any given time, the vector denoting all possible UAV actions is given by $a = (a_1, \ldots, a_\eta)$, and represents all possible combinations of movement available to all UAVs.

As a result, we formulate the immediate reward to all UAVs at time step t as a function of the expected number of people saved due to UAV observation: we require the UAVs to locate not just areas of high population, but areas of population where death rates are likely to diminish the number of survivors at subsequent times.[5] Specifically, this is the product of the expected number of people in a cell multiplied by the death rate in that cell, summed over all unique cells where a UAV is present (i.e. all members of the set G):

$$R(t) = \sum_{G(t)} \bar{p}_{xy}(t) \times \bar{d}_{xy}$$

Over an infinite time horizon, we consider the sum of expected rewards for each time step:

$$\sum_{t=0}^{\infty} R(t) = \sum_{t=0}^{\infty} \sum_{c_{xy} \in G(t)} \bar{p}_{xy}(t) \times \bar{d}_{xy}$$

Since G is itself dependent on the trajectory of each UAV (i.e. the cell each UAV occupies at any time t) this is equivalent to:

$$\sum_{t=0}^{\infty} R(t) = \sum_{c_{xy}(t) \in \mathcal{T}(\mathbf{T})} \bar{p}_{xy}(t) \times \bar{d}_{xy} \tag{4.5}$$

We note here the implications of determinism of detection of people in our model. Whilst the construction below relies on expected values of reward for the exploration of a cell—since we cannot, in advance, know the true conditions on the ground (implied above in the unknown quantities p_{xy} and d_{xy})—we can still consider our MDP model deterministic, with the following justification. All planning in our decision making processes can only rely on the expected value of cell reward. Once a cell has been visited, and the true reward discovered, further exploration of the cell in our model yields no further reward (see below). Furthermore, we do not consider correlation between adjacent cells. As a result, planning is not affected upon discovery of the true reward of visiting a cell, and we can therefore consider the prediction of future expected reward deterministic. We return now, to the discussion of the solution in the context of the available actions for each UAV.

[5] Intuitively, there is very little to be gained from a UAV observing a large group of survivors in an area where there is no danger. We acknowledge the potential ethics of balancing a small group of people in a large amount of danger with a large group of people in a small amount of danger, but the resolution of this dilemma is beyond the scope of this thesis.

Finally, we formulate the standard MDP Bellman optimality equation as defined previously in Equation 2.5, where we elect to denote R's dependence on the actions and state space (which themselves depend on t):

$$Q(a,s) = R\left(a,s\right) + \sum_{s'} P\left(s' \mid a,s\right) \max_{a'} Q\left(a',s'\right)$$

In our case, we do not require the diminishing-returns term γ to ensure $Q\left(a,s\right)$ is finite, since the diminishing value of examining a cell further in the future is expressed in the reduction of the expected number of people as a result of the death-rate term; which is incorporated in the reward function. In other words, $Q\left(a,s\right)$ cannot exceed the number of people alive in the space, given s.

We also do not require the explicit sum over various states since the transition probability P between states given a fixed action is (as discussed above) deterministic; thus reducing the summation term to unity. While this simplifies the form of the action value function, we still require the result be optimal according to:

$$Q(a,s) = R\left(a,s\right) + \max_{a'} Q'\left(a',s'\right) \tag{4.6}$$

from which we obtain the optimal action: $a* = \arg\max_a Q$ to maximise current and future reward. While ostensibly a simplification of a typical MDP formulation, the problem is far from trivial as the joint action space a grows as an exponent of the number η of UAVs in the system, namely:

$$\|a\| \propto a_{\kappa}^{\eta} \tag{4.7}$$

We address this issue in the following section, where we outline our solution.

4.2 Solution Formulation

In this section, we address the problem defined above by introducing a novel multi-agent planning algorithm, based on the MCTS algorithm described in Chapter 2. This is done to address the shortfall in the work in Chapter 3 in addressing Criterion 3 of Section 1.1; calling for *multiple* explorative UAVs searching the disaster space. A naïve approach to such a problem would be to create a joint action tree between all UAVs in the search space. In this scenario, each tree node would represent a combination of actions available to all UAVs in the search-space at each time step. Such a large joint-action solution is, in practice, computationally intractable since it suffers an exponential branching factor with additional UAVs—namely of order a_{κ}^{η}.

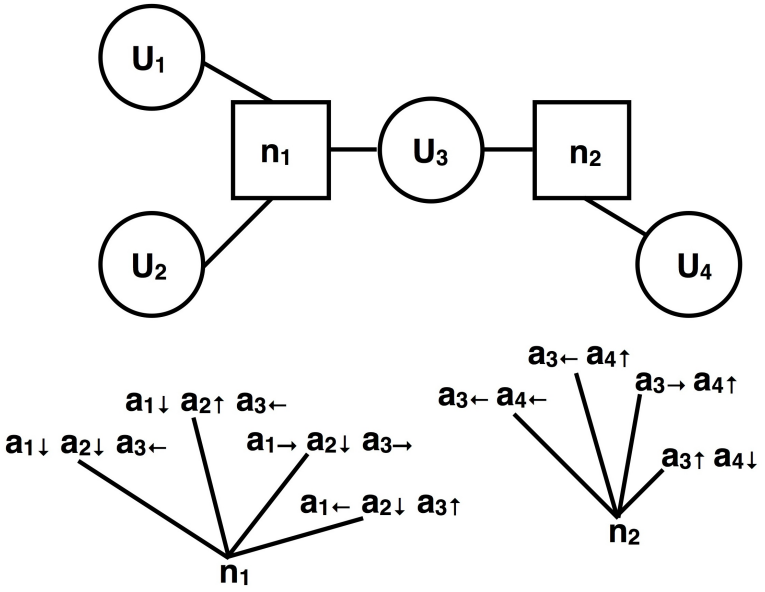


FIGURE 4.1: Example factor graph and corresponding action-space trees created from four UAVs.

Since a regular MCTS would seek to visit each node of this vast search-space at least once, a novel multi-agent path planning algorithm can be introduced using factored joint actions: the *Coordinated Monte Carlo Tree-Search* approach or Co-MCTS. In more detail, we extend the basic MCTS growth procedure (introduced in Algorithm 4) by the creation of joint-action trees between neighbouring (spatially close) UAVs. This approximation of local interaction means the search space is suitably restricted to allow for computationally tractable solution selection. In order to coordinate between local trees, we employ a max-sum implementation during the growth stage of the trees to both synchronise and optimise future planning. In the section below we outline in detail how the joint action trees are generated, expanded, and used to select the optimal action for each UAV.

4.2.1 The Coordinated Tree Search Algorithm

In choosing to exploit spatial locality in UAV interactions, we simplify the search-space considerably by only coordinating between spatially proximate UAVs at any given time. This reduces the size of each joint action tree, and means that in a real scenario the onboard computers of the UAVs can be computing different trees, in parallel, within the team. Recalling that having multiple UAVs search the same cells in the space offers no additional benefit, we can summarise the coordination as ensuring that locally, each UAV has a distinct area to search in the space. As such, the first step of the coordination algorithm is to determine which sub-groups of UAVs need to coordinate their actions at a given time, by working out whether they could occupy the same cell in the search space in the subsequent time step. We now outline the formation of these sub-groups.

Algorithm 8 Joint action graph creation

$\mathcal{J}(G)$

1. $\mathbf{N} = \emptyset$
2. **for** u_κ **in** $\mathcal{U}_h$
3. $this\mathbf{N} \leftarrow \{u_\kappa\}$
4. //For each other UAV from κ to the last UAV n//
5. **for** $u_{\kappa+i}$ **from** u_κ **to** u_η
6. //Spatial separation heuristic//
7. **if spacebetween** $(u_\kappa, u_{\kappa+i}) \leq 2$
8. append $(this\mathbf{N}, u_{\kappa+i})$
9. **endif**
10. **endfor**
11. append $(\mathbf{N}, this\mathbf{N})$
12. **endfor**
13. **Return N**

The coordinated MCTS algorithm begins by calculating which agents require coordination with their neighbours, leading to the form of the agent-based factor graph constructed in the joint-action creation function $\mathcal{J}$ (Line 2), detailed fully in Algorithm 8. The purpose of this stage is that the resulting groups of UAVs will form the basis of the factor graph used in the max-sum calculation (Line 14 in Algorithm 9). The result of $\mathcal{J}$ is represented by a set $\mathbf{N} = \{n_1, \ldots, n_{\kappa<\eta}\}$ that represents the domain of the function nodes to be coordinated. Specifically, each member of $\mathbf{N}$ contains a set of actions corresponding to a group of UAVs that require coordination. In more detail, for some set of neighbouring UAVs—for example $\{u_1, u_2, \ldots, u_\kappa\}$—the possibility exists of $g_1(t+1) = g_2(t+1) = \ldots = g_\kappa(t+1)$ at the next time step $t+1$ of a simulation.[6] In this case, the corresponding element in $\mathbf{N}$—say n_i— would be the set of actions available to these UAVs: $n_i = \{a_1, a_2, \ldots, a_\kappa\}$. Notice that since a UAV may interact with more than one neighbour, the condition $\bigcup_{n_\kappa \in \mathbf{N}} n_\kappa = G$ must hold, whereas $\bigcap_{n_\kappa \in \mathbf{N}} n_\kappa = \emptyset$ will, in general, not. Trees are grown for each n_κ in $\mathbf{N}$, each of which in turn represents the factors in the max-sum graph connected to the variables representing the available actions of the UAVs. Growth is performed Δ times.

An example situation is detailed in Figure 4.1, demonstrating four UAVs $\mathcal{U}_h = \{u_1, \ldots, u_4\}$ of which the first three and the last two may—at the next time step—occupy the same space. Thus the two resulting factor nodes have domains belonging to the set $\mathbf{N} = \{n_1, n_2\} = \{\{a_1, a_2, a_3\}, \{a_3, a_4\}\}$ and form two trees representing joint actions, which are to be grown concurrently. In the figure, four expansions of each initial node have been made representing available combinations of actions performed by each agent connected to that factor node.

The creation of the parent nodes representative of these factors can be seen in Line 6 in Algorithm 9. Following this, the creation and growth of branches is performed inside the loop beginning at Line 8. This begins by exploring down each tree, starting from the parent node, to determine which node to branch on next. Similar to the regular MCTS detailed in Chapter 2, this begins by selecting nodes which have hitherto not been fully expanded: that is, there remain neighbouring action states that have not yet been branched to previously. The total number of neighbouring actions ν from a given action node is of order $\nu = \prod_{a_\beta \in n_\alpha} \mid a_\beta \mid$ for a tree corresponding to factor node n_α, simplifying to $\nu = \mid a_\beta \mid^{|n_\alpha|}$ when all UAVs in the set have the same number of available actions.[7] Thus, the operation of the function:

$$\text{fullyexpanded}(node) = \begin{cases} True & \text{if } e = \nu \\ False & \text{otherwise} \end{cases}$$

[6] We note here a slight abuse of notation, since these nodes serve as *functions* within a factor graph rather than simply a set of actions. Since we factor locally, the functions depend only on the actions in each n and so we omit the function notation for clarity.

[7] We use $\mid x \mid$ on any set x to denote cardinality.

Algorithm 9 Coordinated MCTS

//MCTS Coordination framework
$CoMCTS\,(G, \mathcal{S}, t = 0)$

1. **for** t **in** $[1, \ldots, t_f]$
2. $\mathbf{N} \leftarrow \mathcal{J}\,(G)$
3. $\mathbf{N}_r \leftarrow \emptyset$
4. **for** n **in N**
5. //Root nodes recorded//
6. append($\mathbf{N}_r$) $\leftarrow$ createnode (n)
7. **endfor**
8. **for** *eachstep* **in** $[1, \ldots, \Delta]$
9. //Record nodes to be expanded, and nodes already expanded, beginning from root//
10. $nodestoexpand \leftarrow \mathbf{N}_r$
11. $nodesexpanded \leftarrow \emptyset$
12. **while** $nodestoexpand \neq \emptyset$
13. //Return of max-sum function dictates actions to perform as per Section 4.2.2 (example below)//
14. $actionstoperform \leftarrow$ maxsum $(\mathbf{N}, nodestoexpand)$
15. **for** *eachnode* **in** *nodesexpanded*
16. $rolloutaction \leftarrow$ actionforthisnode $(actionstoperform)$
17. performaction $(rolloutaction, eachnode)$
18. **endfor**
19. //Expansion of tree nodes– similar to standard MCTS//
20. **for** *eachnode* **in** *nodestoexpand*
21. $nodeaction \leftarrow$ actionforthisnode $(actionstoperform)$
22. $newnode \leftarrow$ createchildnode $(eachnode, nodeaction)$
23. //Node considered fully expanded if all available actions expanded on (subject to boundaries of exploration area)//
24. **if fullyexpanded** $(eachnode) = True$
25. remove $(eachnode, nodestoexpand)$
26. append $(eachnode, nodesexpanded)$
27. **endif**
28. **endfor**
29. **endwhile**
30. //Rollout policy carried out as per Algorithm 10//
31. **for** *eachnode* **in** *nodesexpanded*
32. rollout($newnode$)
33. backpropagate($newnode$)
34. **endfor**
35. **endfor**
36. **for** $eachfactor$ **in N**
37. //Best first action selected according to highest reward //
38. **Return** (**bestactions** $(eachfactor)$)
39. **endfor**
40. **endfor**

where e is the number of previous expansions of that node.

Node selection for expansion and growth is then performed using the factor graph as an input for the max-sum algorithm (Line 14).

The rollout portion of the MCTS is traditionally a coarse estimate of the effect of future actions as the result of exploring a particular node in the action space. In this example, we base the rollout on a random-walk through the action space starting at the node just expanded, biased in the direction of the last action taken. This method has the benefit of showing not just the contribution of any random series of actions, but of taking more actions similar to the one represented by the frontier node (for each UAV). Intuitively, a purely random rollout from one node in a joint action tree will be insignificantly different from a rollout from any similar node. Conversely, our rollout policy contributes to the exploration value of a node by indicating possible future reward through continued tree expansion with a preference for repetitions of the action itself. In spatial terms, the rollout explores a random path biased in the direction of the action node expanded upon, yielding future reward information about the spatial region in that direction.

We design the algorithm to plan on-line and re-calculate the optimum action at each time-step. In this way we have no requirement of a priori knowledge of future coordination requirements (as well as built-in robustness to temporal changes in the map), since UAVs re-plan according to their local neighbours each time step. Although not tested for explicitly, we note that this approach ensures communications between UAVs need not be excessive. We note existing literature has shown at-length that max-sum in particular is robust to low bandwidth and irregular message-passing (Delle Fave et al., 2012a; Farinelli et al., 2014; Rogers et al., 2011). In practice we envisage that where a group of UAVs share a tree, a single member of that group will handle the growth and planning of the joint actions.

In order to better outline the functionality of the coordination and growth of the joint-action trees, we now give an example of a max-sum coordination between two trees; followed by the method of expansion of the tree.

4.2.2 Max-Sum Action Selection Example

Consider three agents $\{u_1, u_2, u_3\} = \mathcal{U}_h$ (simplified from our example earlier for ease of comprehension) each able to perform an action to move in a certain direction $a_\kappa = (\uparrow, \rightarrow, \downarrow, \leftarrow)$ where the index κ corresponds to the index of the agent. We consider joint actions between agents 1 and 2, and 2 and 3 represented by the cartesian products of the agents' action sets: $a_1 \times a_2$ and $a_2 \times a_3$. The utility of these joint actions are represented by function nodes in the factor graph in Figure 4.2: $\{n_1, n_2\} = \mathbf{N}$ respectively.

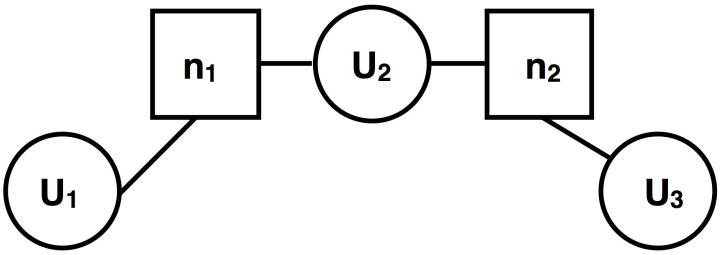


FIGURE 4.2: A simple factor graph used in the max-sum coordination example below. Here two joint action trees (n_1 and n_2) are maintained for three agents (u_1, u_2 and u_3).

Algorithm 10 Rollout

//Performs rollout random walk biased in direction of node's expansion
Rollout(*node*, *direction*)

1. *biasvalue* ← 0.3
2. *prevposition* ← *node.position*
3. *averagereward* ← *node.reward*
4. **for** i **in** $[1, \ldots, steps]$ **then**
5. //Check for bias threshold//
6. **if random**$[0, 1]$<*biasvalue*:
7. //Keeps moving in same direction//
8. *newdirection* ← *direction*
9. **else:**
10. *newdirection* ←select-uniform-probability (u, d, l, r)
11. //Updates reward on the node rolled out//
12. *newposition* ← position $(newdirection, prevposition)$
13. *newnode* ← node $(newposition)$
14. *averagereward* ←updateaverage $(averagereward, newnode.reward)$
15. *True* ← *newnode.seen*
16. *prevposition* ← *newposition*
17. **endfor**
18. **Return** $(averagereward)$

To illustrate a simple example of max-sum coordination in this situation, we show a simple case where the action space is restricted to two actions per agent: $a_\kappa = (\rightarrow, \leftarrow)$. We further define the function node utilities as the following:

n_1	$u_1 \rightarrow$	$u_1 \leftarrow$
$u_2 \rightarrow$	1	2
$u_2 \leftarrow$	4	1

and

n_2	$u_2 \rightarrow$	$u_2 \leftarrow$
$u_3 \rightarrow$	3	6
$u_3 \leftarrow$	1	4

such that (for example) the utility for n_1 should u_1 move right and u_2 move left would be 4. In practice, the values of the utilities would be calculated by the total expected number of people to be found in the cells reached by the selected joint actions (as per Equation 4.5). We can alternatively express this as a pair of vectors:

$$n_1 = \begin{bmatrix} 1 \\ 4 \\ 2 \\ 1 \end{bmatrix} \text{ and } n_2 = \begin{bmatrix} 3 \\ 1 \\ 6 \\ 4 \end{bmatrix} \tag{4.8}$$

Here the elements in each vector correspond to joint actions of: $(\rightarrow\rightarrow, \rightarrow\leftarrow, \leftarrow\rightarrow, \leftarrow\leftarrow)$ respectively (for instance, the third element of n_2 represents the utility of agent 2 moving left, and agent 3 moving right).

Referring to the standard messages passed in max-sum coordination (Section 2.1.3), we define messages passed from function nodes to variable nodes:

$$r_{\beta\rightarrow\alpha}(a_\alpha) = \max_{\mathbf{a}_\beta \setminus \alpha} \left(n_\beta(\mathbf{a}_\beta) + \sum_{\alpha' \in N(\beta)\setminus\alpha} q_{\alpha'\rightarrow\beta} \right) \tag{4.9}$$

where (as usual) the sum is performed over the received messages from all connected nodes except the one being messaged. Here we denote function indices with β and variable indices with α (as before). As such, a_α represents the available actions of agent u_α, and $\mathbf{a}_\beta$ represents the available action vector for the other variables connected to utility n_β. Finally we note the function $N(\beta)$ describing the neighbouring variables to function n_β.

We then recall that the messages passed from variable nodes back to function nodes is defined as follows:

$$q_{\alpha\rightarrow\beta} = \sum_{\beta' \in N(\alpha)\setminus\beta} r_{\beta'\rightarrow\alpha}(a_\alpha) \tag{4.10}$$

In this example the messages are initialised with zeros passed from the variables (agents) to the function nodes as vectors. Beginning with these, the following demonstrates the messages sent and received in this example:

1. $q_{1\to1} = (0, 0)$
2. $q_{2\to1} = (0, 0)$
3. $q_{2\to2} = (0, 0)$
4. $q_{3\to2} = (0, 0)$
5. $r_{1\to2} = (2, 4)$ This represents the utilities of the best two options from the vector n_1 for each possible action from u_2.
6. $r_{2\to2} = (3, 6)$ This represents the utilities of the best two options from the vector n_2 for each possible action from u_2. In this instance, the choice is between the maxima of outcomes for $\to\to$ and $\to\leftarrow$, and $\leftarrow\to$ and $\leftarrow\leftarrow$.
7. $q_{2\to1} = (3, 6)$ Nominally a_2 sums its received messages from every node except the recipient to send onwards. In this instance, it simply has to relay the message previously received from n_2 onwards to n_1 since only these nodes are connected.
8. $r_{1\to1}\,(10, 7)$ Here, the function node n_2 has summed its utility vector to the previously received message, and relayed to u_1 the maximum utilities for each of u_1's available actions.
9. $q_{2\to2} = (2, 4)$
10. $r_{2\to3} = (10, 8)$

At this stage, all three variable nodes have received messages from their neighbouring function nodes and—upon summing these messages—obtain their local portion of the objective utility function z_α (as in Equation 2.4). The decision to take actions $\to$ or $\leftarrow$ can then be made by taking $\arg\max_{x_\beta} z_\alpha\,(x_\beta)$:

$z_1 = (10, 7)$ selecting $\to$

$z_2 = (5, 10)$ selecting $\leftarrow$

$z_3 = (10, 8)$ selecting $\to$

Referring to the notation in Line 14 of Algorithm 9, we note that in our code we return the optimal actions from the Max-Sum calculation as a vector with each element corresponding to each UAV, from which can then be selected each agent's corresponding action in the subsequent lines in the algorithm. Additionally, small amounts of random noise were introduced where tie-breaking was necessary—although in practice, because

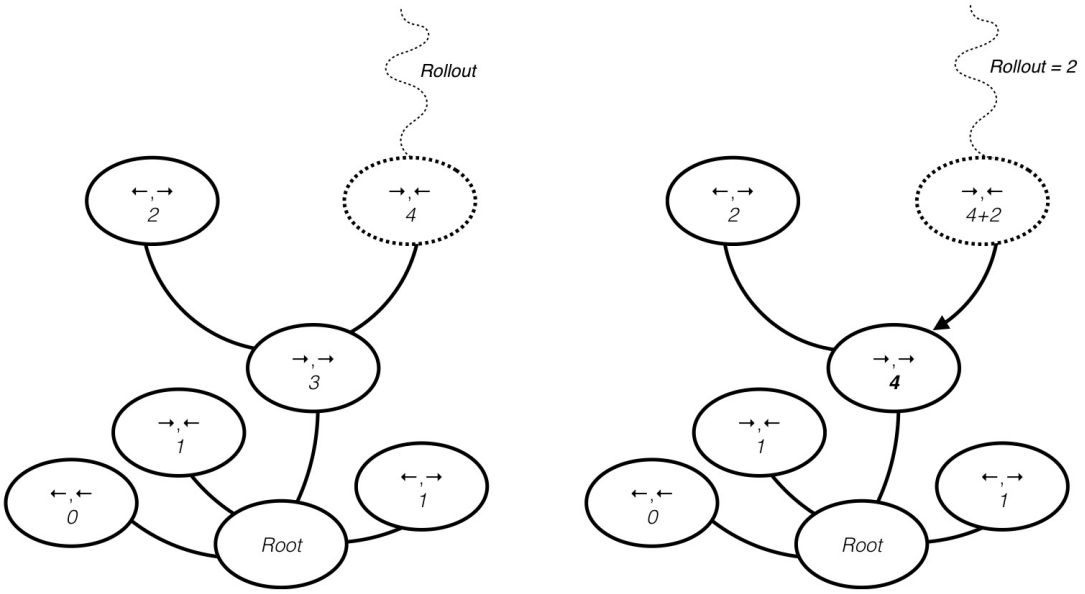


FIGURE 4.3: Example tree expansion for a two-agent joint action tree; considering actions $a = \{\rightarrow, \leftarrow\}$. Once the new node has been added to the tree a rollout is performed as per Algorithm 10, and the resulting value is summed with the node's immediate reward. This value is then backpropagated to the parent node $\rightarrow, \rightarrow$, which updates its value by keeping a moving average of all its child nodes, and its own reward.

of the floating point values representing expected number of survivors, this was very rarely observed.

We further remark that as per our discussion in Section 2.1.3, the max-sum algorithm can be run in parallel and without any synchronicity between nodes with regards to the sending and receipt of messages. Furthermore—as per Section 2.1.3—the message overheads are very small and the calculations simple enough that we typically observed convergence almost instantaneously in simulation.

4.2.3 Tree Growth Example

Following the coordination between joint-action trees detailed above, the addition of the selected node to the tree is carried out. The local reward function of the action is added to the returned value of the rollout policy returned by Algorithm 10, and then backpropagated through each parent node towards the tree root. Subsequent nodes update their own reward value by maintaining a moving-average of its own local reward, and the rewards of any children nodes upstream.

Once growth has been performed to the required tree depth, or according to a maximum set of iterations, the best sequence of actions can be selected by moving down the tree, and selecting the action node with the highest value each time. We allowed for flexibility in the forward planning by re-growing the tree after taking an action to allow re-combination

for new local agent interactions (i.e. recomputing Algorithm 8) while still accounting for future action rewards when considering each move.

Having described the form of our solution algorithm and given some examples, we now discuss the experiments carried out using our algorithm, and the implications of the results.

4.3 Results and Evaluation

We performed experiments to validate the effectiveness of our Co-MCTS solution against a standard un-coordinated MCTS search (where each UAV makes its own decision without regard for the others in the team: i.e. each UAV maintains its own action tree), and a standard "lawnmower" type coverage sweep algorithm (uncoordinated, and with randomised starting locations). These approaches represent standard alternatives to covering a space for the purpose of providing visual information on the situation on the ground, with an un-coordinated approach representing individualistic UAVs (in current disasters, this can act as a proxy representation of individual human operators) and the sweep pattern as a simple frequently-used method for covering a space (as discussed in Section 2.3.1). We run simulations in a centralised fashion—insofar as they are performed on a single computer—but with multiple parallel threads representing the different individual calculations for each portion of the factored utility. In addition, we note that the nature of the max-sum coordination is such that UAVs are not required to have perfect information: it is sufficient that they know their local utility and are able to share this with local neighbours.

We remark again briefly that there are no benchmarks that represent coordinated exploration via tree search for us to compare against directly. Because of the large overheads and computational intractability discussed in Section 4.2, we have not considered a complete joint action tree for these tests. An un-coordinated MCTS search was expected to perform comparably well to the Co-MCTS in situations where the UAVs were spatially separated and therefore already unlikely to need to coordinate their actions. A standard "lawnmower" sweep search is a very common and efficient way to cover a space quickly (Li et al., 2011), but it lacks any kind of directing to areas of higher danger or expected population. An illustration of the lawnmower arrangement is shown in Figure 3.2. As such, we expected performance to generally fall short of both the MCTS and the Co-MCTS except in situations with a very uniform belief map, when it would likely perform better because of the lawnmower's ability to quickly cover the entirety of a space.

In principle the starting locations of the UAVs could have an impact on the final results: particularly in the sweep-search algorithm. As such we uniformly randomised these positions around the edge of the disaster space in each experiment below. One might

expect that, in a real scenario, UAVs are likely to be launched from a handful of operator bases, or a single location. As such we test both scenarios.

Additionally, as per our discussion above on the implications of determinism in our model (Section 4.1), we note here that the metric used in experiments below was of the sum of expected people seen (i.e. "saved") over the entire simulation, averaged over the length of the simulation and number of UAVs:

$$\frac{1}{t_f \cdot \eta} \sum_{t=0}^{t_{f-1}} \sum_{c_{xy} \in G(t)} \bar{p}_{xy}(t) \tag{4.11}$$

where t_f denotes the final time step (the stopping time), and we recall that η is the total number of UAVS and $G(t)$ is the unique positions of the UAVs at time step t. This metric allows us to acknowledge the efficacy of the distribution of exploration among UAVs; as well as to test consistency in discovery across the simulation. We note that, as mentioned in Section 4.2, the justification of using expected values rather than integer numbers of people discovered is because the Co-MCTS method does not use feedback from its observations of numbers of people to inform future belief decisions, since we make the approximation that population is uncorrelated in sufficiently large cells.

The results of various experiments are discussed in the following sections, with detail of the belief map environment used in each case and a discussion of the implication of results.

4.3.1 Results in Artificial Simulated Environments

Initial tests were carried out in simulated environments with damage and population maps created from Gaussian mixture distributions (these being common continuous distributions of unknown variables) : firstly to demonstrate the efficacy of the Co-MCTS method in a controlled environment model and secondly to determine the next-best benchmark against which to measure the performance of Co-MCTS in environments based on real-world data.

4.3.1.1 Performance of Exploration Methods with Varying Danger Position Certainty

Initially, the Co-MCTS algorithm was evaluated in an experimental environment designed to check its performance in situations with varying certainty of the locations of danger. As well as a sweep search the algorithm was decomposed to remove respectively the coordination, the rollout simulation, and the tree growth (i.e. a greedy algorithm that retains the max sum coordination but plans a simple one-step lookahead). This

was done to highlight the contribution of each component of the algorithm to the final performance of the planner. To this end, the environment was composed of a uniform distribution of expected population and a single symmetrical bivariate Gaussian distribution over expected danger, centred in the middle of the search space (composed of a 100×100 grid). Over three experiments the spread of the distribution was increased from five to fifteen sigmas, to examine the algorithm's response to less concentrated regions of danger. Explicitly, using standard notation for a multivariate Normal function with σ selected from $\{5, 10, 15\}$:

$$\bar{d}_{xy} \doteq \mathcal{N}_2 \left(\begin{bmatrix} 50 \\ 50 \end{bmatrix}, \begin{bmatrix} \sigma^2 & 0 \\ 0 & \sigma^2 \end{bmatrix} \right)$$

We set $t_f = 3000$ and repeated each experiment 1000 times, taking our uncertainty from statistical standard-error.

The results are shown in Figure 4.4, with the metric used being the average number of survivors discovered per time-step of the simulation, per UAV (consistently set here at 4 to allow coordination complexity, but still remain fast to compute; with examination of differing numbers of UAVs in Section 4.3.1.3 below). The Co-MCTS algorithm consistently outperformed a non-coordinated tree search, a tree search without any rollout, a greed max-sum implementation, and the sweep search under all three conditions. Broadly it was shown that performance decreases across the algorithms as the danger area becomes more diffuse across the belief map; empirically because it becomes harder for the UAVs to reach the population in danger promptly if that area of danger is larger. Because the coordinated Monte Carlo Tree-Search targets first the population with the highest likelihood of death, it maintains a more consistent value of people found. In contrast, while the MCTS also possesses this "targeting" ability, it falls short of the coordinated approach in situations where the UAVs are in close quarters with each other, and likely to move towards the same area. In our problem formulation, multiple UAVs observing the same cell either simultaneously or concurrently offer no benefit since the value of $\bar{d}_{xy}$ reduces to zero when $o_{xy} = 1$, as per Equation 4.3, and we avoid double-counting the expected number of people in each cell.

Since Co-MCTS uses both coordination and forward simulation of explorative paths, the results for the number of people found per UAV in the simulation per time step are consistently closer to the maximum value (that is, 1 person) than the other methods examined, supporting our assertion that this algorithm is effective and fast at finding people in danger in a disaster situation. In particular, the temporal advantages of forward planning is especially apparent in comparisons to a greedy (but nonetheless coordinated via max-sum) policy. This is of particular interest since previous work has shown that there are problem scenarios where coordinated greedy max-sum is a good solution (La Cesa et al., 2008; Tisdale et al., 2009; Yedidsion et al., 2014).

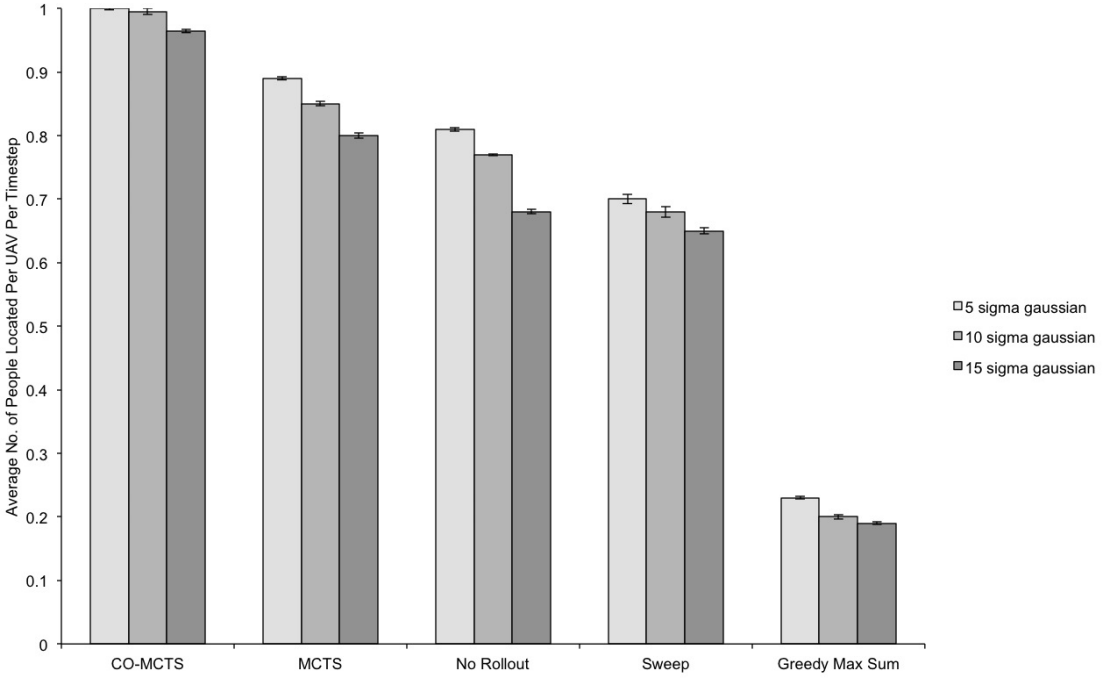


FIGURE 4.4: Performance results for the Co-MCTS algorithm, uncoordinated MCTS algorithm, and a standard lawnmower sweep search over a belief map consisting of a uniform expected population and a single Gaussian component forming the expected death rate, centred in the middle of the search area. The increased sigma value of the Gaussian component reflects a more widely spread region of expected danger.

Having established the viability of the Co-MCTS algorithm, we now examine the response of this method to changes in the form of the belief map to show consistency across different distributions of belief data (such as would be encountered in differing disaster locations).

4.3.1.2 Performance of Exploration Methods with Varying Belief Map Complexity

We examined the relationship between increasing numbers of Gaussian components forming the danger element of the belief map and the efficacy of the three exploration methods. Specifically we examined this behaviour to give us an insight into the behaviour of the Co-MCTS planner in more realistic situations; which are typically more random than a single Gaussian component (Venanzi, 2014). Therefore, it was important to establish that adding further components to create a more complex danger belief map did not remove the advantage of using our coordinated planner as opposed to the uncoordinated and sweep-search methods. Positions of the Gaussian components of danger were randomised uniformly across the grid in each experiment. As before, we simulated 4 UAVs, set $t_f = 3000$ and repeated each scenario 1000 times, recording standard errors to ensure statistical significance. It was expected that the exploration of a completely

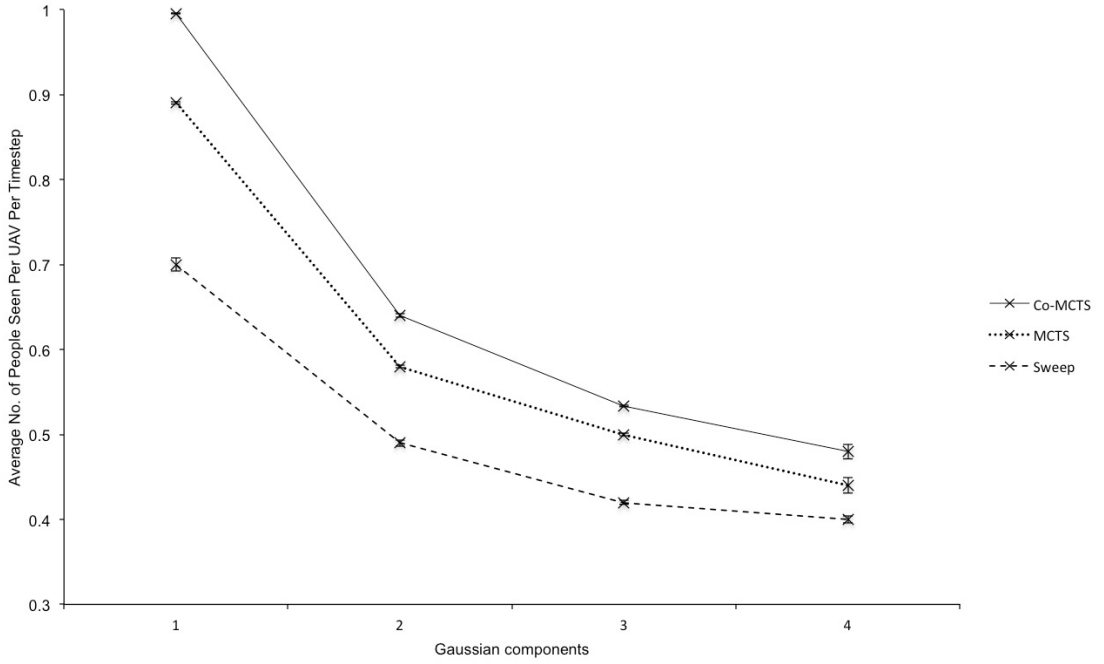


FIGURE 4.5: Performance results for Co-MCTS, MCTS, and lawnmower sweep for varying number of Gaussian components forming the basis of the expected death rate. Sigma was fixed at $\sigma = 5$ throughout. Results show an expected decline in average people seen per UAV per time step with increased components to the belief map, although the Co-MCTS algorithm maintained consistent quality benefits to the other algorithms examined.

uniform map of danger and population distribution would be best performed by a sweep-search, since no single area of the map requires more examination than any other. The results presented in Figure 4.5 support our assertion as—although the performance of the Co-MCTS algorithm decreases in an approximately asymptotic manner with additional components—the Co-MCTS remains the most effective choice up to at least four components by approximately 10% compared to a sweep search.

Thus far, we have used four UAVs to carry out the exploration of the search space. In the next section we examine the dependence of the performance of each algorithm on the number of UAVs in the scenario.

4.3.1.3 Performance of Exploration Methods with Varying Numbers of UAVs

We next examined the performance of the Co-MCTS algorithm with the addition of further UAVs to the exploration scenario, in fulfilment of Requirement R3 outlined in Section 1.1. The expected performance of the coordinated algorithm was a higher number of people discovered per UAV per time step than the two non-coordinated methods since in the former method each UAV acts in a complementary way in order to increase total people seen. Thus, as shown by the results in Figure 4.6, we found no statistically significant decrease in people discovered per UAV per time step from 1 to 6 UAVs in the

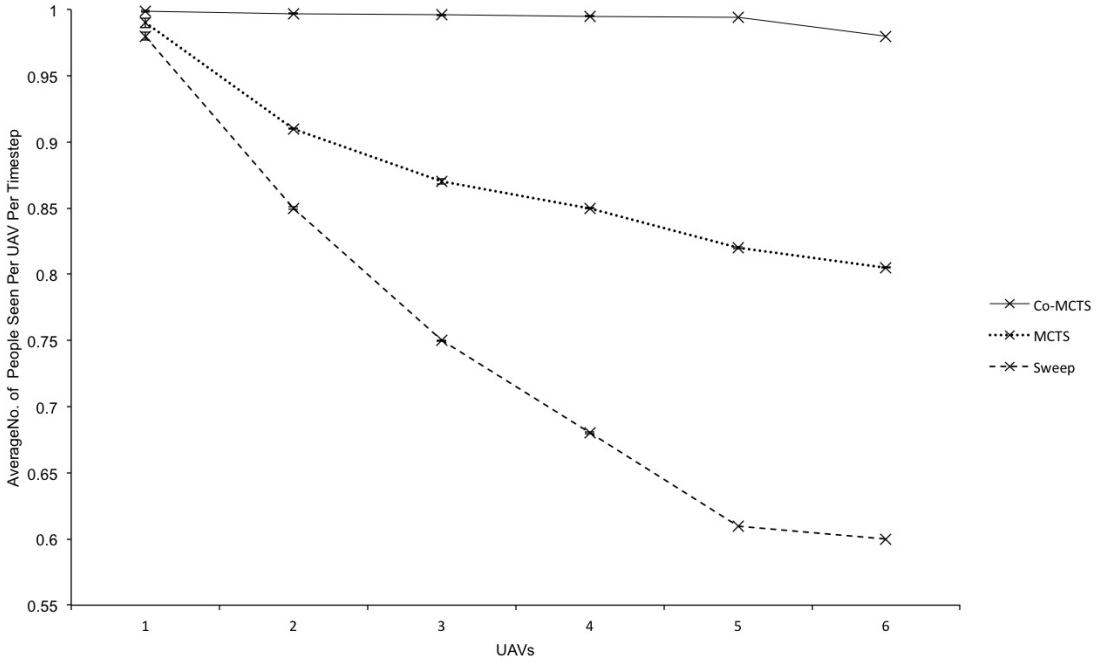


FIGURE 4.6: Performance results for Co-MCTS, MCTS, and lawnmower sweep for varying number of UAVs in the environment, over a 10σ Gaussian distribution. Results confirm no statistical decrease in the average number of people seen per UAV per time step in only the Co-MCTS case, reinforcing the evidence that the UAVs are working collectively to maximise rewards for the whole team.

simulation ($t_f = 2000$ and 1000 repeats of each scenario). Conversely, MCTS and the lawnmower sweep search demonstrated more substantial (almost 30% in the latter case) decreases in result quality as additional UAVs that did not coordinate their actions were more likely to search overlapping areas, reducing the overall number of individuals seen per time-step.

4.3.2 Results in Environments Generated from Ushahidi Dataset

To explore the performance of our algorithm on data relevant to real-world disaster scenarios, we used data from the Ushahidi project (Morrow et al., 2011) produced from crowd-sourced information during the 2010 Haiti earthquake.[8] The results from these data are published in Baker et al. (2016a).

In more detail, we constructed a *decomposed*—or discretised—grid world of size 200×200 of $10m$ cells, with UAVs traversing from the centre of one cell to the centre of an adjoining cell above, below, or to either side at each time step. This is convenient since assuming a UAV speed—typical of quad rotor vehicles—of $10ms^{-1}$ amounts to the traversal of one cell in one time step of one second.

[8] Available from `http://www.ushahidi.com/`.

Specifically, we extracted the level of damage and coordinates of buildings in a 2km square centred on the capital, Port-au-Prince. Damaged buildings represent an estimate of the damage in an area and thus, the danger to the victims on the ground: buildings that have suffered more severe damage will likely lead to more severe casualties and a higher rate of death. We formed a belief map of danger to the populace by summing the total number of buildings above a threshold level (set to a crowd report of damage 3 and above on the Ushahidi crowd reporting scale from 1-5) in each cell, before multiplying by a common factor to convert the data into a map representative of expected fatalities (noting the constraint in Equation 4.1). The environment is displayed in Figure 4.7 with a scale showing the value of d in each location.

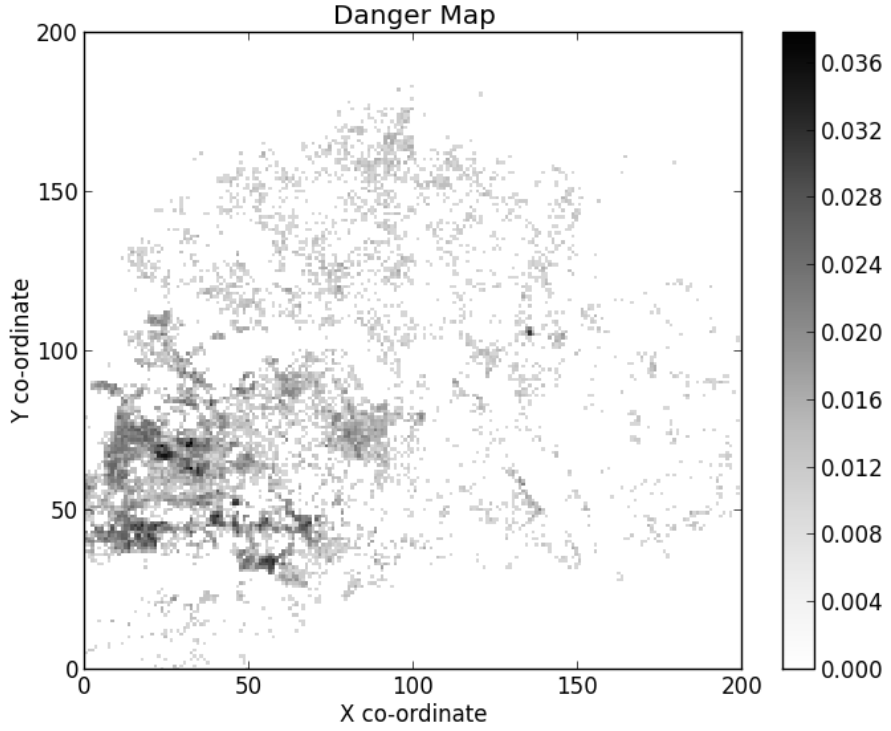


FIGURE 4.7: Danger d_{xy} shown spatially, created from Ushahidi dataset centred over Port-au-Prince. Dimensions of $2km$ along each side.

The number of unique sets of UAV positions G (that is, the cardinality of the state space) in this environment for (for example) five UAVs, is of the order 8.5×10^{20} referring to Equation 4.7; even without the variation in the expected number of people with time.[9] This is in contrast to earlier work in this field that only dealt with state spaces with hundreds or thousands of states available (Amato and Oliehoek, 2015). As such our work represents the first application of coordinated tree-search to the type of large action domains more reflective of real-world scenarios: essential for Criterion 3 of our goals (in Section 1.1) requiring the algorithm be suitable for belief maps of the type used in disaster response.

[9] Calculated using standard multichoose combinatorics (Feller, 1968).

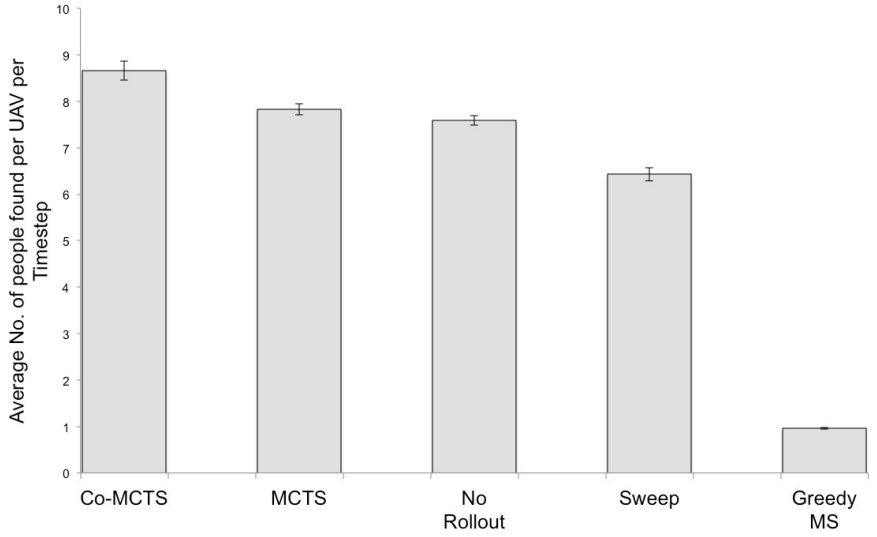


FIGURE 4.8: Comparison of performance between: coordinated MCTS, un-coordinated MCTS, simple lawnmower sweep search, Co-MCTS with no rollout policy, greedy policy with max-sum coordination. Start locations were uniformly randomised, $\eta = 4$, $t_f = 1000$.

In the experiments below, we use the performance metric defined above in Equation 4.11, averaging over the length of the simulation and the number of UAVs.

4.3.2.1 Initial Performance of Exploration Methods in Ushahidi Data Environment

Figure 4.8 shows our results for an initial test of the coordinated Monte Carlo tree search using randomised starting locations for four simulated UAVs. We compare against, a "lawnmower" sweep search as before, as well as a typical MCTS algorithm without the factored coordination. We also demonstrate explicitly the forward planning required in the survivor discovery problem, by benchmarking against Co-MCTS without a rollout policy, and a greedy (but locally coordinated via max-sum) policy. Our results demonstrate a minimum performance increase of 10% over MCTS, and higher gains over the other benchmarks. Errors are taken as standard error of the mean over 1000 repeats of each experiment. The poor performance of a greedy one-step lookahead shows that even with coordination, the ability to forward-plan to account for survivor death rates is essential to discovering casualties in our scenario. Indeed, simple un-planned lawnmower-style sweep searches are more successful, but still fall short of the MCTS's ability to plan exploration paths to target (for instance) high-danger areas before casualty rates become too high.

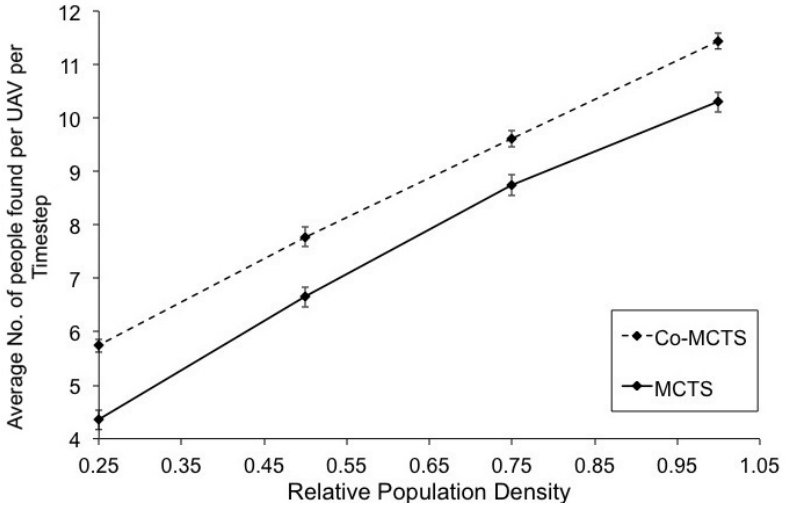


FIGURE 4.9: Comparison of coordinated and un-coordinated MCTS in locating survivors over varying densities of population, showing Co-MCTS's consistency in different forms of belief data. Here $c_{0,0} = [0, 0]$; $t_f = 1000$; and $\eta = 5$.

4.3.2.2 Performance of Exploration Method with Varying Belief Map Density

For the algorithm to be viable, it should show performance benefits in a variety of situations with a variety of distributions of casualties to be discovered, as already indicated in Section 4.3.1.2. We achieve this by varying the relative population over the belief-space from 1 (as above), down to 0.25 by uniformly sampling from the Ushahidi dataset to the population portion of the belief map and comparing to the most successful benchmark from our initial simulation (uncoordinated MCTS). This has the effect of maintaining our standard of using real data with a large action space, while still allowing us to experiment over different state spaces (S). The results presented in Figure 4.9 demonstrate this consistency with an average 14% improvement over the un-coordinated MCTS benchmark: even at low densities of people; where the coordination requirements would be intuitively less valuable. We thus provide evidence that the performance gains of Co-MCTS are not simply a by-product of the test environment selected.

4.3.2.3 Performance of Exploration Methods with Varying Numbers of UAVs in Ushahidi Data Environment

Additionally, in line with Section 4.3.1.3 we explicitly demonstrate the benefit of the *consistency* afforded by our coordination in a simulation. We do this by varying the number of UAVs present and comparing to the closest benchmark to Co-MCTS's performance from our initial simulation. Good coordination should maximise the overall reward gained by the algorithm (namely the number of people discovered) with low diminishing returns compared to uncoordinated approaches. This is demonstrated in

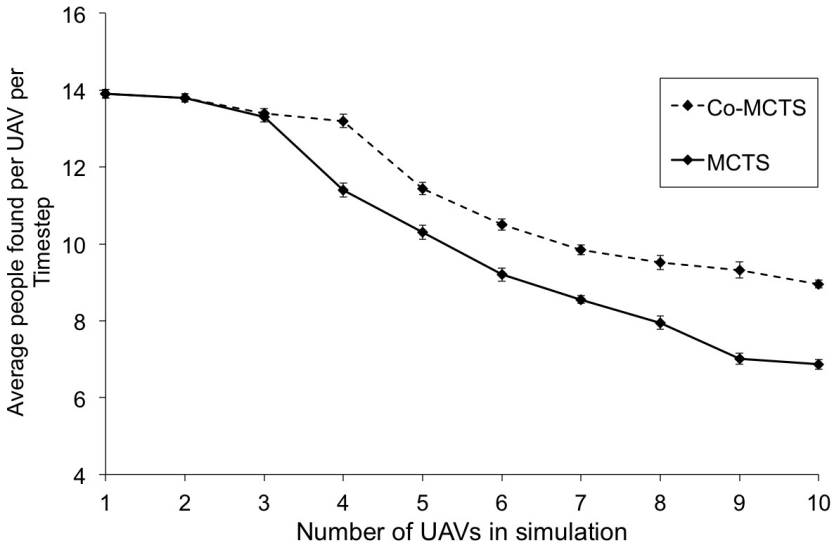


FIGURE 4.10: Comparison of coordinated and un-coordinated MCTS in locating survivors with additional UAVs; demonstrating a more consistent performance per-UAV in Co-MCTS. Initial UAV starting locations fixed at $c_{0,0} = [0, 0]$; $t_f = 1000$.

Figure 4.10 for varying η, where we note performance improvements of over 23% with the inclusion of 10 UAVs (and an average of 11% across the experiment). This is notable after 3 UAVs are introduced, since before this point interaction (and therefore required coordination between the UAVs) was at a minimum since the search space was large enough to accommodate multiple non-intersecting explorative paths. For more UAVs, coordination is more commonplace and with the successful implementation of Co-MCTS there is higher value in the inclusion of further UAVs into the scenario, compared to a situation without coordination.

4.4 Summary

In this chapter, we have summarised our construction of the first coordinated factored MCTS algorithm—a novel coordinated exploration algorithm—and its efficacy in three experimental scenario models, and three scenarios based on real-world data from reports of damaged buildings in an earthquake disaster area. Specifically, this work was done as a logical extension of the combined mechanism presented previously in Chapter 3 enabling multiple explorative UAVs to participate in a disaster response scenario: directly addressing the shortfall of our previous work in meeting research Criterion 3 (multiple explorative UAVs) in Section 1.1. In particular we developed an algorithm in order to fulfil Research Requirements R1 and R3 calling for decentralised autonomy and scalability respectively; by allowing the addition of further UAVs in the exploratory role (considered only for a single UAV in Chapter 3). In more detail, our new algorithm extends into the exploration domain the idea of decentralised decision making between UAVs; in this

case for the construction of explorative paths; while allowing for the scalable use of more UAVs than the initial single-explorer method presented in Section 3.2.2.

We acknowledge that work in the area of factored coordinated tree-searches has been performed by Amato and Oliehoek (2015)—developed simultaneously and independently from our own algorithms—and that the principal differences lie in the coordination of the factored trees. Specifically, Amato and Oliehoek apply a simple variable elimination algorithm whereas we use the max sum coordination algorithm detailed above. Moreover, the action space in our example is large (of order 10^{20} states) in comparison to the example chosen by Amato and Oliehoek; who select two model problems with state space cardinalities of 4, 16, 243, and $\sim 180,000$. Neither problem involved the recalculation of joint trees as the simulation progressed, in further contrast to our problem.

We have verified via experiment that the coordinated planner outperforms non-coordinated MCTS and a coverage algorithm over a variety of belief-map environments by discovering more survivors per UAV per time step. We have also shown how the Co-MCTS algorithm maintains consistent returns for additional UAVs in the scenario as a result of coordinating their search actions. In the following chapter, we extend the algorithm into a domain where the decomposition of the actions of the UAV into simple orthogonal movements is not required and we present the first example of a *continuous* coordinated tree-search.

Chapter 5

The Continuous Space Coordinated Path Planning Mechanism

In the previous chapter we introduced an application of coordinated factored planning to address the problem of multiple UAVs discovering survivors in a disaster area (research Criterion 3). In doing so, we made a number of assumptions about the form of the planning; specifically that the actions taken were selected from a finite discrete set of the cardinal directions across a grid. This contribution therefore constitutes a *discrete* planning algorithm.

In this chapter, we introduce and discuss an extension to this work allowing for coordinated planning over *continuous* action spaces, and how we apply this to a UAV search and rescue scenario. In the following sections we first describe our motivation for extending the problem by removing the discretisation assumption (Section 5.1), in order to best acknowledge the range of motion available to UAVs and extend our scenario model to better reflect the type of sensors deployed on these vehicles. Next, we introduce the specific formulation of the environment model and UAV behaviour considered in our simulations (Section 5.2), before introducing the continuous form of our coordinated Monte Carlo tree-search algorithm (Section 5.3). Following this, we present empirical evidence to substantiate the benefits of this approach (Section 5.4), before summarising our findings (Section 5.5).

5.1 Extension of Coordinated Path Planning into a Continuous Action Domain

The work described in Chapter 4 requires some degree of simplification before planning can commence (Section 4.1). Specifically, this involves discretising the environment into a number of *cells* to be examined. Since locations in the real-world are not discretised in

this way, this requires some additional processing of incoming data before UAVs can begin their exploration. Furthermore, by simplifying data in this way it is inevitable that some control over planning is lost when a UAV can only be considered as visiting single cells, rather than being able to plan according to a *continuous* range of motion. Moreover, sensors on explorative UAVs are more complex than is suggested by a simple observability boolean variable (denoted o_{xy} in the previous chapter); we instead introduce a more realistic sensor model reflecting the kind of instrumentation that could be deployed in a disaster space.

In this chapter we seek to address these shortcomings in planning UAV searches of disaster areas. By so doing, we make the following contributions to the state of the art (contributing to meeting research Criteria 1 and 3):

1. We introduce an extension to the survivor discovery problem introduced in Section 4.1; specifically modelled on a likely real-world scenario with the goal of locating an unknown number of people, over a wide area, by detection of mobile phone signals and where the diminishing survivability of the people with time is incorporated into the reward function. This approach removes several limiting assumptions regarding the otherwise simplistic sensor model that (as outlined in Section 4.1) previously simply observed the number of survivors in a cell with perfect accuracy.

2. We extend the algorithm introduced in Section 4.2.1 to allow multiple UAVs to plan explorative paths over a *continuous* action space, reflecting a more realistic range of movement for the vehicles and improving the performance of the coordinated explorers. Our approach utilises the aforementioned formulation of the discovery problem and belief maps of the locations of mobile phone signals to generate planned paths that do not rely on any decomposition of the action-space available to the UAVs.

3. Furthermore, in order to demonstrate the applicability of our scenario to potential future disasters, we test and evaluate our approach on the same real-world data (gathered from the 2010 Haiti earthquake) from Section 4.3, showing consistent gains in survivor discovery of at least 7% compared to the discretised coordination algorithm, with higher gains of around 20% for scenarios with additional UAVs.

Applying coordinated tree-search to a continuous actions requires significant changes to the existing approach of Chapter 4, both in the formulation of the scenario, and in applying sampling of the continuous action-space while still allowing UAVs to coordinate with one another. Crucially, discrete approaches to MDP solutions require some form of recognition that a particular set of actions has been entirely explored: clearly where a continuum of actions exists this cannot be said to be true, since the number of individual actions available for selection is infinite. As such we have had to extend different

approaches usually applied to planners dealing with continuous spaces (Coulom, 2006; Wang et al., 2008) into the regime of coordinated exploration.

5.2 Continuous Exploration Scenario Model

Having outlined the motivation for extending our algorithm beyond the work in Chapter 4, we begin by discussing the conceptual changes required when moving from a discrete to a continuous model. Nevertheless, we note that the high-level aim of our work is allowing first responders in disasters to minimise the loss of human life; an aim our previous scenario formulation (Section 4.1) explicitly acknowledges by combining data about a region containing known hazards (for example, high levels of radiation or presence of fire) and regions where it is known or believed that there are likely to be persons in the vicinity. As a result, this combination of factors—and the belief maps created from danger and population data—remain the focus of this extension to our algorithm.

However, in moving to a continuous range of motion for UAVs we must discard the notion of "cells" which can be considered visible (or observed) or otherwise, which had previously featured in our environment model (Section 4.1). Moreover, while making our algorithm applicable to a continuous space, we also take the opportunity to address our assumption that survivors in the scenario can be considered completely "discovered"—and thus located—with absolute certainty. Specifically we introduce a more realistic sensor model with a related utility value where planning can focus on localising the probable position of survivors as far as possible.

To clarify the differences between this new scenario and that outlined in Chapter 4, we will now outline the formulations describing the new sensor model, the reward gained from exploration, and the range of motion available to UAVs.

5.2.1 Continuous Space Environment Model and Reward Function

As we have indicated, in order to move away from the discrete model in Chapter 4 we must first discard the notion of deterministically observing a cell, and instead consider realistic indicators of the presence of survivors in a given location. Consequently, we exploit the ubiquity of mobile phone ownership and assume the use of mobile phone signals as proxies for the presence of a person. As well as having precedent in previous use in disaster scenarios (Goetz et al., 2011; Zorn et al., 2010), this has the specific advantage of allowing identification of individual sources using unique identifiers associated with each handset. While a priori knowledge of the number of victims in a given area might be unavailable, first responders can maintain a belief distribution over unobserved victims while also attempting to isolate signals that have been observed, in order to reduce the uncertainty in the location of victims.

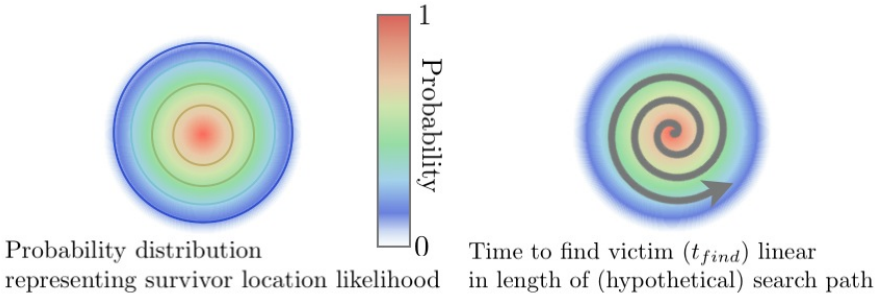


FIGURE 5.1: An example of the relationship between the uncertainty in the location of a survivor, and the hypothetical time taken to find them using a sweep or spiral search. It is desirable to localise (as far as possible) the locations of victims so that when first responders attend to their expected position they minimise the time taken to locate them; ensuring higher survivability.

As a result, we explicitly envisage a scenario where UAVs are equipped with some form of detector capable of providing a (noisy) estimate of the range of individual unique phone signals. Specifically, we seek to localise the expected position of victims in order to reduce the time taken by search and rescue teams to find (and subsequently rescue) them (see Figure 5.1). In more detail, we associate the uncertainty of a person's location with the time taken to search the area for that person. By moving the detectors around the space, the expected location value can be determined with higher precision; effectively reducing the area to be covered (and thus the time taken) by rescue services from a large initial area to a much smaller location.

We consider a search area containing a number of signals $s \in \mathcal{S}$ indicating the presence of people in some *danger* (mapped spatially by a two dimensional scalar function $\mathcal{D} : \mathbb{R}^2 \mapsto [0, 1]$), corresponding to their expected likelihood of dying within the next time-step t. Each signal s_i is mutually distinguishable from other signals, and the magnitude of each can be sensed by UAVs within a set radius. The reward we gain R (replacing Equation 4.5 in Section 4.1) is related to the number of people we hope to observe, their likelihood of survival, and a discovery time t indicating how long it would take to rescue any victim:

$$R = \sum_{p_i, d_i \forall s_i \in \mathcal{S}} p_i \cdot (1 - d_i)^t$$

where $d_i \in \mathcal{D}$ and p_i represents the expected number of people for a given signal $s_i \in \mathcal{S}$. We adopt the index i from Chapter 3 since we can label individual signals as "incidents".[1] In the first instance we assume a relatively flat prior belief of victim position, implying a long time to locate an individual. However, with a set of observations (O) of—for example—the strength of a mobile phone signal some estimate can be made of the location of a person; effectively reducing the time to locate them. We denote this using a *time*

[1] In particular, such incidents could then conceivably be assigned to the low-level task responder UAVs of the type discussed in Chapter 3, in order to (for example) deliver medical supplies or return live imagery to first responders of the casualties.

to find parameter t_{find} that decreases linearly with the estimated area of the location of an individual phone signal.

As such, at any given time our reward is then:

$$\sum_{p_i,d_i \forall s_i \in \mathcal{S}} p_i \cdot (1 - d_i)^{t_{find}(O)}$$

whereas projecting reward to any arbitrary time in the future we have for a series of observations at time t:

$$R_{known} = \sum_{p_i,d_i \forall s_i \in \mathcal{S}} p_i \cdot (1 - d_i)^{t_{find}(O_t)+t}$$

By collecting information on signal sources we can use a population Monte-Carlo (PMC) (Cappé et al., 2004) to model the likely locations of a person. This approach functions as an iterative importance sampler, which increases in precision as more measurements are taken—explicitly reducing t_{find}. Thus, reward (R) is a fundamentally a function related to the danger at a given location believed to contain a victim, and the precision with which the location of that person is known. This intrinsically encodes the logical notion that casualties in more danger should be localised more accurately, to enable emergency responders to locate them quickly. Planning can be performed by simulating the result of measurements on the probability distribution for each person and extrapolating the effect on t_{find} in each instance.

To account for signals we have yet to observe, we include a term for the *expected* number of victims outside of the range of observations. In principle the search area for such victims would be the entirety of the area over which observations have yet to be recorded, since the positions of the victims are known with no localisation whatsoever. Thus for a continuous distribution of expected people ($\bar{p}$):

$$R_{unknown} = \int \bar{p}_{xy} \cdot (1 - d_{xy})^{t_{find}} \, dxdy$$

In practice we approximated this function by estimating the total number of people yet to be sensed by the UAVs using the belief maps of danger and population, and treating the time-to-find as the time to explore the entire un-sensed region. In other words we assume a hypothetical discovery time based on exploring the entire as-yet unseen region of the map.

The global reward function at time t then simply becomes the sum of the two components:

$$R_{total} = R_{known} + R_{unknown} \tag{5.1}$$

It is worth noting that since the formulation of reward from the population Monte-Carlo simulation is essentially separate from the tree-search function outlined below, the two are (in general cases) separately applicable.

5.2.2 UAV Behaviour Formulation

Having outlined the new continuous space environment, we now introduce the extensions to the action space of the UAVs themselves, and the formulation of their continuous range of motion (replacing our previous model from Section 4.1). We consider simple UAV dynamics—including minimal constraints on performance—since the focus of our work is on planning rather than constraint optimisation, and because restrictions on UAV behaviour can be included in subsequent iterations of the model as constraints on the reward function . Thus the set of UAVs $\mathcal{U}_h = \{u_1, \ldots, u_\eta\}$ traverse the space in iterations of a fixed distance v per time-step t (i.e. at fixed speeds and altitudes); with a continuous domain of available angles available to determine the direction of the next action.[2] The action vectors enabling UAV u_κ to move at the next time step is defined as $a_\kappa = \left(a_\kappa^\alpha, a_\kappa^\beta, a_\kappa^\gamma, \ldots\right)$ where each Greek index can be interpreted as an angle between 0^o and 360^o. In theory the cardinality of a_κ is infinite, but as detailed below we use continuous space tree-search methods to restrict our search to finite subsets sampled from the full range. Each UAV selects a sequence of actions to produce its trajectory $T_\kappa = [a_\kappa(t=1), a_\kappa(2), \ldots, a_\kappa(t_f)]$ (for a trajectory that ends at time t_f); which together form the set of all trajectories $\mathbf{T} = \{T_1, \ldots, T_\eta\}$. Thus the collective goal of the UAVs is to plan a set of trajectories to satisfy: $\mathbf{T}^* = \arg\max_{\mathbf{T}} R(\mathbf{T})$.

5.3 The Coordinated Continuous Monte Carlo Tree-Search Algorithm

In retaining Monte-Carlo tree search as the basis for our solution, we recall its ability to sample very quickly from large state spaces (traditionally used in solving games), and the flexibility with which it can be applied to general problems, including in continuous action domains (Browne et al., 2012; Couëtoux, 2013). As described in Section 4.2, we exploit locality between UAVs to factor the search space into local joint-action trees. Consequently, we allow trees to coordinate over shared factors (that is, shared UAVs) using the max-sum algorithm by exchanging messages to express the local reward gained by UAVs taking particular actions at future times. Functionally this is similar to the example given in Section 4.2.2, except with a mechanism in place to sample from the continuous actions available rather than selecting from a finite list of cardinal directions.

[2] Since the length of time in a time step t is defined below as $1s$, and the velocity of the explorative UAVs is $v\left[ms^{-1}\right]$, the distance travelled in a single time step is simply $v\left[ms^{-1}\right] \times 1\,[s] = v\,[m]$.

We now outline the changes made to the Co-MCTS algorithm from Chapter 4. Our contribution constitutes a modification of the *selection* and *expansion* steps of the Co-MCTS process of tree growth and coordination. We recall that MCTS is typically summarised: node *selection*, *expansion*, *rollout* or simulation, and *backpropagation* (Section 2.3.4.1). In the previous chapter, we modified the selection process to determine which node to expand by coordinating in parallel between trees via max-sum (Section 4.2.1): we now additionally allow the algorithm to sample from the available continuous actions. The creation of the trees proceeds in identical fashion to the procedure in Section 4.2.1, and the rollout portion is functionally identical to that outlined in Algorithm 10, with the modification in Line 10 to allow uniform sampling from the full continuous range of directions in the new scenario.

In our continuous approach, algorithm 11 begins with the creation of the root nodes representative of each factor seen in Line 7, which are recorded in the set $\mathbf{N}_r$ (Line 4). Following this, the creation and growth of branches is performed Δ times inside the loop beginning at Line 9. This begins by exploring down each tree, starting from the root node, to determine which node to branch on next. Node selection is performed in accordance with Polynomial Upper-Confidence Trees (PUCT) (Coulom, 2006; Wang et al., 2008) introduced in Section 2.3.4.2. We recall that this approach outperforms similar earlier continuous space planners, and has a main drawback (cycles between states) not applicable to our scenario—where UAV actions directly and irreversibly alter the state of the observations of the environment.

In detail, we sample uniformly and randomly from the action set of a node up to a limit of K actions per node, with K defined as in Couëtoux et al. (2011a) to be a parameter of constants $C > 0$ and $\alpha \in (0, 1)$ and the time of simulation t: $K = Ct^{\alpha}$, with α defined as a function of the depth of the node ε:[3]

$$\alpha = \frac{1}{10(\varepsilon_{max} - \varepsilon) - 3}$$

for $\varepsilon \leq \varepsilon_{max} - 1$ in line with the discussion we presented in Section 2.3.4.2.

Line 11 introduces the current set of nodes (across all trees) to be expanded next, $\mathbf{N}_{next}$, and Line 12 creates the set of previously expanded nodes $\mathbf{N}_{prev}$. At Line 15 the max-sum algorithm is used to maximise the value of rewards (as per Equation 5.1) the actions over each n_ι, returning a vector of favourable actions $a^* = (a_1^*, a_2^*, \ldots, a_\eta^* \mid a_\kappa^* \in a_\kappa)$. Since each n_ι depends on a subset of actions, the function select$(n^{(\kappa)}, a^*)$ serves to return only the actions corresponding to a given $n^{(\kappa)}$. This is then used as the argument to create the new expansion to a node in $\mathbf{N}_{next}$ in Line 18.

[3]We choose ε as opposed to the d used by Auger et al. (2013) and Couëtoux et al. (2011a) to differentiate from the nomenclature used in describing danger functions and distributions.

Algorithm 11 Continuous Coordinated MCTS

$CoCMCTS\left(\mathcal{U}_h, \mathcal{D}, t=0\right)$

for t **in** $[1, \ldots, t_f]$
  //Creation of factor graphs given UAV locations//
  $\mathbf{N} \leftarrow \mathcal{J}\left(\mathcal{U}_h\right)$
  $\mathbf{N}_r \leftarrow \emptyset$
  for n_ι **in** $\mathbf{N}$
    //Root nodes//
    append$(\mathbf{N}_r) \leftarrow n_\iota^{(0)}$
  endfor
  for *eachstep* **in** $[1, \ldots, \Delta]$
    //Node to expand//
    $\mathbf{N}_{next} \leftarrow \mathbf{N}_r$
    $\mathbf{N}_{prev} \leftarrow \emptyset$
    while $\mathbf{N}_{next} \neq \emptyset$
      //Max-sum coordination returns best actions for shared factors//
      $a^* \leftarrow \text{maxsum}\left(\mathbf{N}, \mathbf{N}_{next}\right)$
      for $n^{(\kappa)}$ **in** $\mathbf{N}_{next}$
        //Actions relevant to $n^{(\kappa)}$ selected//
        $n_{new}^{(\kappa)} \leftarrow \text{expand}\left(n^{(\kappa)}, \text{select}\left(n^{(\kappa)}, a^*\right)\right)$
        //Fully expanded metric//
        if expansions$\left(n^{(\kappa)}\right) \geq K$
          remove$\left(n^{(\kappa)}, \mathbf{N}_{next}\right)$
          append$\left(n^{(\kappa)}, \mathbf{N}_{prev}\right)$
        endif
      endfor
    endwhile
    for $n^{(\kappa)}$ **in** $\mathbf{N}_{prev}$
      //Simulation (rollout) and backpropagation of results//
      rollout$(n_{new}^{(\kappa)})$
      backpropagate$(n_{new}^{(\kappa)})$
    endfor
  endfor
  for n_ι **in** $\mathbf{N}$
    //Best actions selected//
    $a*_i = (\textbf{bestactions}\left(n_\iota\right))$
  endfor
endfor

5.4 Results and Evaluation

To verify the performance of our algorithm on data relevant to real-world disaster scenarios, we used the same data—as in the previous chapter—from the Ushahidi project (Morrow et al., 2011) from the 2010 Haiti earthquake to generate $\mathcal{D}$ and $\mathcal{S}$.[4] As in our previous scenario, damaged buildings represent an estimate of the damage in an area and thus, the danger to the victims on the ground. For the sake of simplicity of evaluation, we constructed a grid of size 200×200 of $10m \times 10m$ cells (as in the previous chapter), to form the basis of the danger function $\mathcal{D}$. This function is displayed in Figure 4.7 with a scale showing the value of d in each location. Once again we assume a UAV speed—typical of quad rotor vehicles—of $10ms^{-1}$. This amounts to planning one action of moving $10m$ in one timestep of one second. We typically simulate UAV searches over time horizons of $t_f = 1000$. As before, we ran simulations in a centralised fashion but with multiple parallel threads representing the different individual calculations for each portion of the factored utility. We envisage that—in practice—leaf UAVs in joint action factor graphs would be tasked with maintaining shared trees in a realistic deployment; or that computation would be distributed according to the processing power of each vehicle (in the event that the explorers are heterogeneous). We discuss these implications further in Section 6.2.3.

In calculating values for use in the reward functions, the value of danger is based on the mean expected position of the signal based on the data collected. Where the spatial location is not yet clearly established we have found that empirically the change in spatial danger was smooth enough that nearby values tended to be close to the final estimated value of d_i in most cases.

In contrast to Section 4.3.2, our performance metric used is the percentage reduction in the time for total cumulative discovery time t_{find} (averaged over the number of UAVs) since it best reflects the ability of the UAV search to pinpoint victims for rescue. We benchmark against a similarly coordinated—but discrete—MCTS implementation identical to the algorithm presented in the previous chapter, but updated with the reward functions from the new environment model. Since we have shown (in Chapter 4) that discrete Co-MCTS is suitable for explorative path planning—and outperforms other approaches (Section 4.3)—this is a sensible choice of benchmark to ensure a continuous algorithm represents equal or better results. As before, the action space of the UAVs is restricted to moving between the cells forming the danger-function environment. This scenario poses similar challenges of coordination in large action spaces but benefits from existing work that deals with factored finite-space tree-search (Amato and Oliehoek, 2015). The results of the simulations outlined below are published in Baker et al. (2016b).

[4] Available from `http://www.ushahidi.com`.

In putting the algorithm into practice we note that the communications overheads were not excessive, with plans taking less than a second where coordination was not required, and only slightly longer where it was. This introduces a potential limit on planning speed since—in general—we assume time steps of one second between actions. We remark on this further in Section 6.2.2.

5.4.1 Initial Performance of Exploration Method in Ushahidi Data Environment

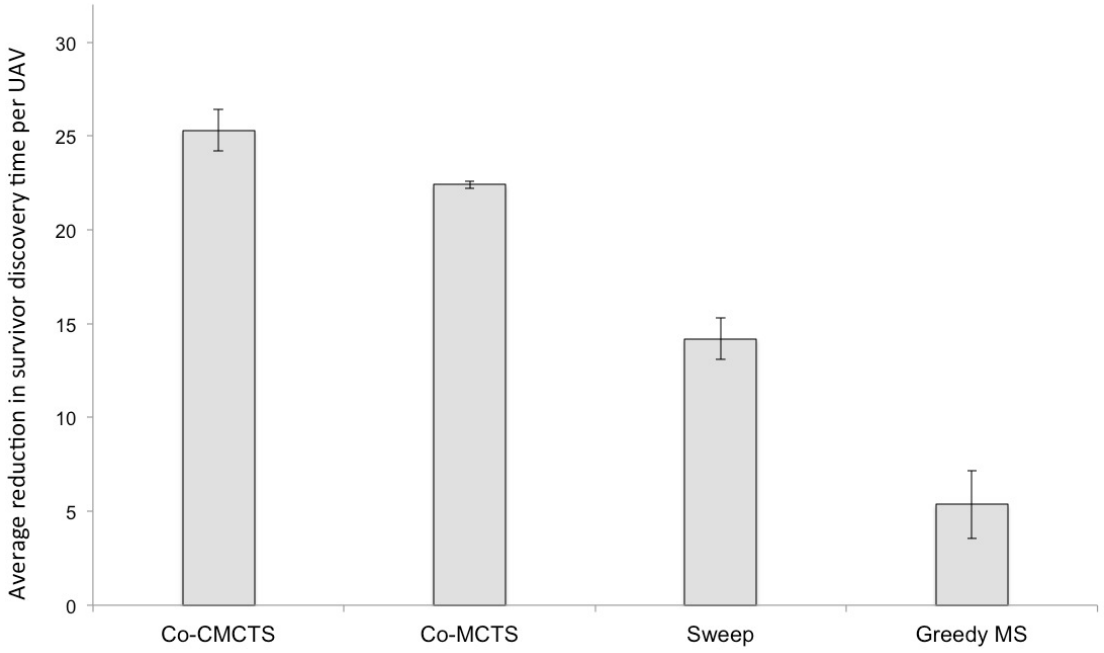


FIGURE 5.2: Result of randomised starting position tests for each of the continuous coordinated MCTS, discretised (cellular) coordinated MCTS and a simple lawnmower sweep-search; performed 10^6 times. Results indicate reduction in t_{find} averaged over the four UAVs in the scenario.

An initial simulation with four UAVs in randomised start locations on the map shown in Figure 4.7 is shown performing against a discretised coordinated MCTS implementation (as described in Baker et al. (2016a)) and a simple lawnmower-style sweep search over the area for comparison. This shows a gain of around ∼7% over a discretised search space (Figure 5.2), highlighting the benefit of not restricting the UAVs to the four cardinal directions when planning actions to ensure survivors are located.

In particular, this benefit is due to the continuum of actions available being less restrictive than in a cellular decomposition of the search area; allowing more effective coordination. Notably, when moving in an up-down-left-right fashion across a space, any travel distance is necessarily equal to the Manhattan distance between start and end locations. Conversely, a continuous action space allows for more direct—and therefore shorter and

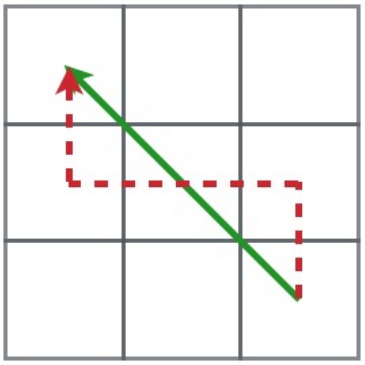


FIGURE 5.3: Simple schematic, showing how a UAV (red dashed line) moving only in the four cardinal directions (up, down, left and right) must travel a longer distance to reach locations not directly above, below, or to either side of its starting position when compared to a UAV with a continuous range of motion (green line).

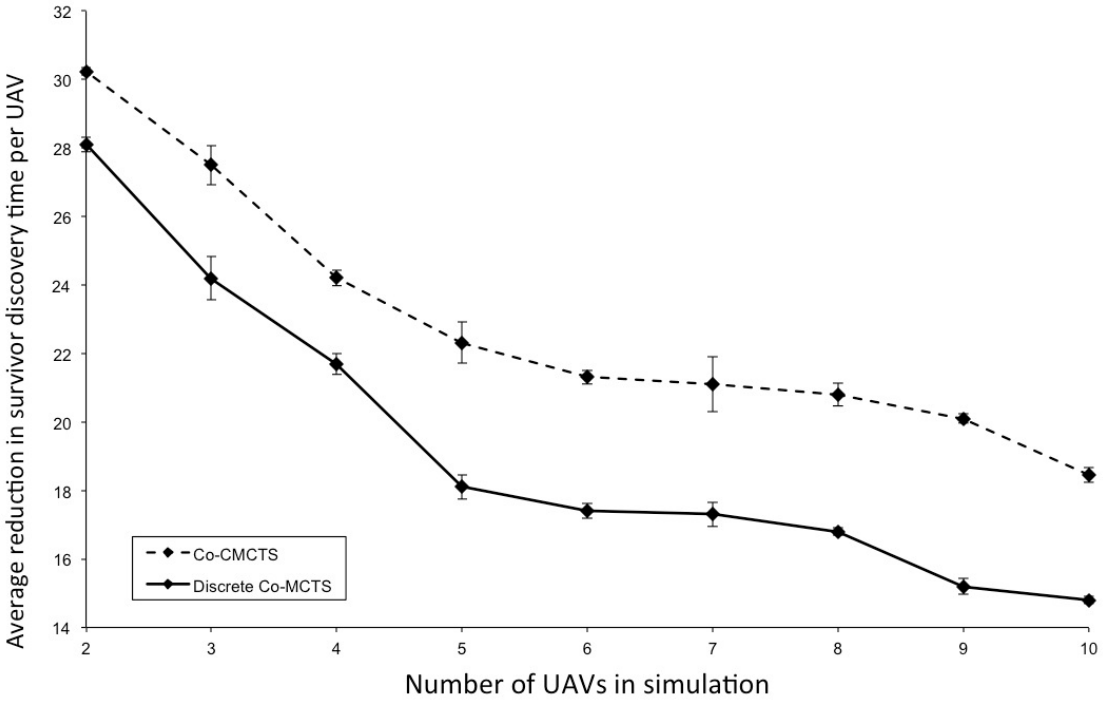


FIGURE 5.4: Comparison of continuous and discrete coordinated MCTS in a $t_f = 1000$ simulation of varying numbers of UAVs. The continuous space approach is not only better than the discretised approximation, it is more consistent in its reward per-UAV added to the scenario. Results here are averaged per-UAV in the simulation.

faster—Euclidean distances to be moved by traversing in straight lines between any two locations. An example of such a scenario is detailed in Figure 5.3.

5.4.2 Performance of Exploration Methods with Varying Numbers of UAVs in Ushahidi Data Environment

Furthermore, we are able to demonstrate the consistency of our approach on addition of further UAVs to the simulation. Intuitively the reward gained by each UAV in a

well-coordinated algorithm should suffer fewer diminishing returns when adding more to the scenario. This is because any additional UAVs should still localise approximately the same number of signals as other UAVs in the environment, if they coordinate the exploration task effectively as a group. If they do not, one would expect additional UAVs would explore the same regions of the disaster space as those already present: which, as discussed previously, offers negligible improvements to the global reward function when compared to localising previously un-seen casualties. We demonstrate in Figure 5.4 that additional UAVs results in a slower decrease in observed reward than in the discretised action space. Most notably at 8 and 10 UAVs the difference in performance per-vehicle is approximately 20% in favour of the continuous algorithm.

5.5 Summary

Motivated by the need to remove limiting assumptions in our previous work, in this chapter we have introduced an implementation of a decentralised, factored, coordinated Monte Carlo tree search algorithm for the purpose of discovering survivors in a simulated UAV path planning scenario by extending the techniques of Co-MCTS into a continuous action regime. Furthermore, we have introduced a more realistic representation of reward (or utility) in the environment by considering a sensor model of a mobile phone detector, such as might be deployed in a genuine disaster. These developments addressed more comprehensively the criterion that our algorithm allow multiple UAVs to perform path-planning while also being a more accurate model of the range of motion available to these vehicles than restricting them to four set directions (Criteria 1 and 3 in Section 1.1). Furthermore, we continued to meet requirements R1 and R2 since our algorithm is both decentralised, and designed to coordinate the actions of the explorers to maximise total reward. Tests were carried out on the same real-world data from the 2010 Haiti earthquake used in Chapter 4. We demonstrated the capability of our Co-CMCTS algorithm to sample this space and planning paths, and demonstrated consistent performance gains over the discretised Co-MCTS algorithm (scaling up to ten UAVs as per Requirement R3) in the localisation of survivors discovered of up to 20%.

Chapter 6

Conclusions and Future Work

In this thesis, we began by examining the various benefits, and issues that arise, with the deployment of UAVs to explore a disaster area and respond to incidents by returning imagery or sensory data to first responders.

- Firstly, we studied existing methods for planning explorative paths, allocating tasks to agents, and calculating predictive prior placement for these agents to minimise later travel time to tasks. We identified several areas of research where we felt work was required to develop existing algorithms and methods to make them more robust and applicable to real-world situations. Specifically, we focussed on the lack of a combined mechanism for exploring and task allocation. Furthermore, we noted a lack of work in the field of cooperative path planning for the purposes of discovering survivors.

- Secondly, we presented a new mechanism for heterogeneous UAVs to explore and respond to incidents in a disaster area, subject to a belief map informed by other organisations or crowdsourced reports on the ground.

- We then extended our work to account for multiple UAVs simultaneously exploring a grid-world, and created a new coordinated tree-search to enable forward planning. This framework also incorporated a more nuanced approach to a belief map, containing information on both expected location of victims and their likelihood of death.

- Finally we removed the assumption of a discretised world (and action space) to create the first continuous coordinated tree-search planner, enabling UAVs to coordinate over a continuum of actions. We also introduced a more principled environment model based on detecting and isolating mobile phone signals, rather than continuing to model survivors as simply "observed" when a UAV is over their location.

We now report our conclusions (Section 6.1) drawn from our work and experiments, followed by a discussion of future work (Section 6.2) we believe forms the basis for further experimentation and ultimately deployment of our algorithms on UAV platforms.

6.1 Conclusions

In Chapter 1, we introduced the background and motivation of using UAVs in disaster relief, as well as our key requirements for our algorithm to fulfil. We focussed on describing a mechanism to allow the discovery attendance of incidents in a disaster area as informed by a belief map. We outlined the goals of having heterogenous UAVs for these roles, that task-attending UAVs could be pre-placed near to possible incident locations, and that the aim of the mechanism was to reduce the time for all incidents to be attended. Furthermore, we imposed the requirements of the mechanism being decentralised, robust, and scalable.

In Chapter 2, we examined the existing literature on path planning, task allocation and predictive placement algorithms critically, while considering the research requirements of our project. Although a significant amount of work has been done towards allowing UAVs to coordinate effectively with first responders in disaster situations, there remained areas that had not been fully developed, which we believed we could improve upon. Specifically, previous attempts at combining the aspects of heterogenous UAVs had not focussed on the incident-location and response form of attending to a disaster despite this being a framework representative of actual disaster relief efforts. Nonetheless, we were able to use some of the literature to inform the subsequent development of our mechanism. We concluded that a modified RRT algorithm for path planning, a max-sum DCOP formulation for task allocation, and a simulated annealing approach for prior placement were the most promising bases from which to construct our work.

Consequently, in Chapter 3, we built on this existing work to create the foundation of the first complete system for exploration of a space for incidents, and task allocation once they have been discovered—the Path Planning, Task Allocation and Placement problem (or PPTP problem). This framework represents the first attempt at unifying exploration with task allocation in this way, and in functioning as a proof of concept, is a significant step towards an operationally ready mechanism to allow UAVs to aid first responders in genuine disaster relief scenarios. As well as demonstrating the efficacy of the mechanism as a whole, we evaluated our modified RRT path planning algorithm and found it to perform better than a coverage method in incident location by roughly 100%. Finally, we used a prior placement algorithm on the task-attendant UAVs with a view to reducing the future time for them to respond to task allocation at incidents. As a result we have achieved the contributions outlined in Section 1.2 to the extent discussed in Section 3.5. Subsequently, we have demonstrated that our algorithm performs better in

our simulated environments than benchmarks with the path planning, task allocation, or predictive placement components of the system replaced with alternative naïve methods. In particular, the time for discovery of incidents, allocation of incidents to UAVs as tasks, and time for UAVs to attend their tasks, was lower using our mechanism compared to other methods except for where the UAVs have perfect information (an unrealistic upper-bound benchmark). We successfully showed that in the experiments conducted, our system performed better than the benchmarks by as much as a 167% reduction in time to locate and attend incidents. Consequently, we have demonstrated the applicability of this set of algorithms to disaster response scenarios where first responders are looking to deploy UAVs for the purposes of exploring a space and attending tasks as they are discovered.

Subsequently, we identified a need to expand the abilities of the exploratory UAVs in our combined mechanism to allow multiple explorers to work together to better find people in danger in a disaster space. To this end, we developed the first coordinated factored Monte-Carlo Tree Search algorithm (Co-MCTS) to coordinate exploration of the search space among multiple agents. Such work is vital in ensuring that casualty numbers are minimised and first responders are maximally informed of the locations of any persons in danger. In experiments we demonstrated the viability of the coordinated MCTS algorithm in situations with varying degrees of complexity in the belief map of the information on the ground, and found consistent benefits even with relatively small numbers of additional explorative UAVs compared to an uncoordinated approach (Baker et al., 2016a). We outlined the relevance of this additional work in the context of the research goals outlined in Section 1.1. Despite the contributions achieved, we concluded that there was further work to be done to ensure the system more accurately reflects real-world scenarios, and fully overcomes the challenges and goals identified in Chapter 1.

As a result, we introduced a more principled model of UAV actions that allowed a continuum of possible movements (rather than discrete actions transitioning from grid-cell to grid-cell), and a corresponding continuous environment model. This was based on the detection and localisation of mobile phone signals corresponding to casualties in the disaster area. Working with the same data as in the discrete case, we developed a new continuous coordination algorithm (Co-CMCTS)—the first example of such—that outperformed the discretised case by as much as 20% (Baker et al., 2016b).

Consequently, we have shown the viability of a combined exploration and allocation framework that could be deployed on UAV platforms, and extended this work to create a new coordinated path planning algorithm that makes use of the kind of belief data available in a real world scenario. We hope that this work leads directly to algorithms that are deployed on platforms whenever a natural or manmade disaster results in a need to locate and attend to casualties: particularly where first responder personnel are stretched and would benefit from an autonomous system without high manpower costs.

The techniques we have demonstrated here have applications beyond disaster response use: including similar UAV deployment for other remote sensing problems. UAVs are seeing use in the monitoring of agriculture (Katsigiannis et al., 2016; Valasek et al., 2016), forestation (Messinger et al., 2016; Wallace et al., 2016), and glaciers (Ely et al., 2016; Kraaijenbrink et al., 2016): all of which are problem domains where—given a group of exploratory vehicles and some prior beliefs about areas to explore—the algorithms developed in this thesis can provide autonomous solutions. Aside from the exploration of a physical space, complex planning of robot component motion (by exploring configuration, rather than geographical, spaces) (Kavraki et al., 1996; LaValle, 2006); and pursuit and evasion scenarios (Bernardini et al., 2015; Chen et al., 2015; LaValle, 2006) are two further fields in which action spaces can be explored via a tree search, and where coordination between multiple components or agents may be required. In both instances a continuous factored tree search could provide a potential solution to otherwise intractable problems of planning.

6.2 Future Work

The contributions we have presented represent an initial step towards a viable real-world system for the exploration of, and response to, incidents in a disaster space using heterogeneous UAVs. In what follows, we detail the key areas that will require the development or improvement of existing algorithms to fully meet the requirements in Section 1.1. We have outlined this required work below in Sections 6.2.1–6.2.3. These focus primarily on removing assumptions, improving performance, and making the system more effective in realistic situations.

We also note briefly now, that the logical conclusion of our algorithm development is the combination of Co-CMCTS with our combined PPTP solution framework; to allow multiple UAV explorers to coordinate and explore simultaneously. Since the task allocation problem—which receives input from any explorative UAVs—is already decentralised, we envisage no significant obstacles to achieving this combination of algorithms. Nonetheless as we have already intimated, the nature of a "task" as a single location requiring a UAV to attend is somewhat simplistic and not representative of real scenarios. This distinction represents the principle obstacle to the combination of our coordinated exploration algorithm with the framework outlined in Chapter 3, since it requires reevaluation of the nature of a "task" to better reflect the requirements of aid workers and the subsequent modifications to the allocation algorithm that would follow.

6.2.1 Removal of Assumptions in Exploration Algorithm

Despite our successes, our work contains a number of assumptions that future work should seek to refine or remove to make the algorithms as realistically applicable as

possible. Most notably, the belief map is assumed to be static (despite the dynamic implications for death-rate discussed in Section 4.1); whereas in reality, information on the possible locations of incidents can be received at any time as the situation progresses, which might affect the form of the belief map by adding or removing components. Future work should test the ability of our system to deal with dynamic belief maps that change form with time to reflect the receipt of new information. Since the use of the path planner is currently robust to dynamism by virtue of the MDP formulation (Section 5.3), we do not envisage problems with continued use of the current path planning algorithm: particularly with prior belief information on the likely appearance of future incidents.

Furthermore, the model does not take into account the possible future development of incidents in regions already observed. In practice, beliefs about the location of potential victims may be updated as the UAVs move around the disaster space. With data on the likelihood of the occurrence of an incident in the disaster space in a unit of time, it would be possible to place distributions over areas of the map previously explored, in order to accurately reflect the probability of further survivors requiring rescue in those locations growing as time passes. For example, if the exploring UAV covers an area and discovers no survivors, the value of the belief map in this area would decrease to zero to reflect this (or near-zero, accounting for sensing uncertainty). Following this, if the UAV is no longer located in the aforementioned area, the belief map probability would increase with time according to some distribution, to account for the likelihood that new victims are discovered in the region since it was last observed.

These approaches will require further study of disaster relief data—particularly regarding observed distributions of survivors and urban population densities—to ascertain realistic probability distributions and sensor models to use in the scenario, since a mathematical framework that does not reflect the genuine experiences of first responders will not prove useful should the system be deployed in a real disaster. Further datasets (such as the Ushahidi Haiti data used in our simulations) should be used to create scenarios with time-varying conditions, as well as for general experimentation to ensure consistent performance across different distributions of survivors in different disaster scenarios.

Finally, we expect that heuristics currently used in the algorithm—for instance the distance at which UAVs coordinate their actions, or exact values of 'danger' computed from crowd sourced reports—can be numerically "tuned" (i.e. adjusted) to increase reward and survivability of casualties. This could be achieved by, for example, examination of additional data and humanitarian reports on UAV performance, survivability rates, and typical sizes or compositions of search environments; or by employing reinforcement learning techniques.

6.2.2 Performance Improvements to the Exploration Algorithm

We note that there remains scope for improving the performance of the path-planning algorithm: both to make use of additional exploration techniques and to increase the prospects of using the algorithm in a real-life situation. In light of this, we envisage further experiments to improve the performance of our algorithm to ensure that, in a disaster relief setting, UAVs can find victims in as short a time as possible. In more detail, there are a number of specific features of our problem of exploration that more general formulations do not contain; which can be used to improve response times without sacrificing any solution quality. Specifically, we have identified three additions to the exploration algorithm that that should be explored in order to best exploit the capabilities of the UAVs:

1. There are a number of options for distribution of computation (most notably of factor graphs and trees) between UAVs that must be explored (Ben-Asher et al., 2008; Chmaj and Selvaraj, 2015; Pascarella et al., 2015). Specifically, experiments are needed to determine the optimal method for sharing of calculation and tree growth during coordination, with which one could minimise computation time and thus generate plans more quickly.

2. By studying data from disaster environments and also from population demographics, models could be generated to reflect likely correlations of survivors in certain areas. As a result, prior beliefs of the locations of civilians can be less uniform and more reflective of the types of buildings or locations being explored. Furthermore, by using the presence or absence of survivors in a certain location, inferences about the likelihood of populations being present in the surrounding area can also be made; resulting in more robust plans. We note that this is a distinct addition to the algorithm from the discussion in Section 6.2.1 regarding future incidents in already-explored locations, although the same data on population densities and distribution would be likely beneficial to the construction of both models.

3. Simulations should be conducted to explore the validity of incorporating—at a local level—the benefits of sweep-search patterns into the explorative path planner. In particular, sweep-searches or spiral-searches are often very effective search policies in small areas (Bernardini et al., 2014, 2015; Zargar et al., 2016): in our scenario, areas of relatively flat belief data. Furthermore, lawnmower sweep patterns are more understandable by first responders; an important consideration should the algorithm ever be developed for end-users with a focus on flexible autonomy (Ramchurn et al., 2015). We envisage a scenario where, in addition to single step-by-step actions for explorative UAVs to take at the next time-step, planning can also account for larger macro actions representing spiral or sweep search policies to be carried out. As long as planning continues at each iteration of the simulation

such a planning policy could still adapt to new information or updated beliefs at later times.

6.2.3 Deployment on UAV Platforms

Ultimately, the goal of simulations and experiments centred on discovering survivors in disaster environments is to produce an algorithm (or set of algorithms) that can be deployed on UAV platforms and subsequently used in the aftermath of a disaster. While such deployment is unlikely to occur in the very near future, we note that the following five developments would form the basis of a logical progression ending with trials on real airframes:

1. A realistic sensor model should be studied to acquire data on the heuristics implemented in simulation (see Section 6.2.1). Specifically, the type of data generated from mobile phone reception sensors (simulated in our continuous survivor discovery problem) should be studied and incorporated into the reward functions of the exploration algorithm as a matter of course, so that when this information is provided by real sensors the paths planned still reflect the best series of actions for localising survivors.

2. We have already indicated in Section 6.2.2 above that the distribution of computation between different UAVs is important for the performance of the exploration algorithm. In addition, considerations must be made of the limitations of computational power available to different airframes, and how this will affect the speed of generation of plans. Some work in this field has been carried out insofar as deploying a max-sum task allocation algorithm on two platforms (Delle Fave et al., 2012b), with promising results. Further work should compare typical UAVs employed in disaster response and examine the available computational power, which may not represent the entire processing power of a UAV if flight systems require substantial input from a CPU controller. Indeed, in order to ensure widespread applicability on a variety of devices with varying computing power, it may be prudent to develop simpler (sub-optimal) algorithms—ideally that interact with the algorithms we have introduced—for use on simpler UAVs.

3. In deploying the exploration algorithm on UAV platforms it would be beneficial to use a versatile set of programming languages and APIs that are suitable for heterogenous platforms. In our combined mechanism description we alluded to the open source Robot Operating System (ROS) package (Section 3.3) as an example of such a software framework. Once the exploration and task allocation algorithms are implemented in such a way the only obstacle to deployment is designing an interface between ROS and the onboard flight-control software on the UAVs themselves.

4. Regarding logistical considerations of a test on real airframes, a suitable area would need to be identified that was either capable of simulating the large spatial extent of a genuine search space, or a to-scale location would need to be found. Although some work has been done (Delle Fave et al., 2012a), the latter option remains difficult due to restrictions on UAV flights by national aviation authorities—at least in the United Kingdom (Civil Aviation Authority, 2016)—and by the limited performance of cheap testable airframes (see below). Consequently the sensing capabilities and sources of mobile phone signals used in our environmental model would need to be scaled either post-signal reception or artificially at a hardware level to compensate for the comparatively smaller ranges between UAVs and the signal sources being detected.

5. Finally, the platforms on which to deploy the algorithm must be decided on. This decision would most likely focus on the practicality of obtaining sufficient UAVs to run a compelling test, while also requiring a minimum performance standard to ensure adequate flight time and movement capabilities. For initial testing, a rotorcraft like the DJI Spreading Wings (described in Figure 1.1) would be most suitable, since rotorcraft are necessarily more agile than fixed-wing drones and thus require fewer constraints on path-planning arising from performance limits. Ultimately, the algorithms should be deployed on as many platforms as practical since there can be no guarantee that the vehicles available in any given disaster will conform to a narrow selection of test UAVs.

Unmanned aerial vehicles are being used increasingly often in the aftermath of disasters, and the demand for further exploitation of their capabilities is high (Baehr, 2015; Measure, 2015; OCHA, 2014). The logical progression of these deployments is a move towards vehicles that operate partially or entirely autonomously from human pilots; and with continued investment and research in this field, such deployments could conceivably become commonplace within a few years.

Bibliography

Ushahidi, 2015, `http://www.ushahidi.com/` (Accessed: 2015-08-24).

S. M. Adams and C. J. Friedland. A Survey of Unmanned Aerial Vehicle (UAV) Usage for Imagery Collection in Disaster Research and Management. In *Proceedings of the Ninth International Workshop on Remote Sensing for Disaster Response*, volume 9, Stanford, MA, 2012.

S. M. Adams, C. J. Friedland, and M. Levitan. Unmanned Aerial Vehicle Data Acquisition for Damage Assessment in Hurricane Events. In *8th International Workshop on Remote Sensing for Disaster Management*, pages 1–7, 2010.

S. Adlakha, S. Lall, and A. Goldsmith. A Bayesian Network Approach to Control of Networked Markov Decision Processes. In *2008 46th Annual Allerton Conference on Communication, Control, and Computing*, pages 446–451, Monticello, IL, sep 2008. IEEE.

B. Adler, J. Xiao, and J. Zhang. Autonomous Exploration of Urban Environments using Unmanned Aerial Vehicles. *Journal of Field Robotics*, 31(6):912–939, 2014.

N. Agmon, G. A. Kaminka, S. Kraus, and M. Traub. Task reallocation in multi-robot formations. *Journal of Physical Agents*, 4(2):1–10, 2010.

R. Agrawal. The Continuum-Armed Bandit Problem. *SIAM Journal on Control and Optimization*, 33(6):1926–1951, 1995.

M. Alighanbari, J. P. How, and L. Bertuccelli. A Robust Approach to the UAV Task Assignment Problem. In *Proceedings of the 45th IEEE Conference on Decision & Control*, volume 18, pages 5935–5940, San Diego, CA, 2006. IEEE.

S. Amador, S. Okamoto, and R. Zivan. Dynamic multi-agent task allocation with spatial and temporal constraints. In *Artificial Intelligence*, pages 1384–1390, Quebec City, Canada, 2014. AAAI.

C. Amato and F. A. Oliehoek. Scalable Planning and Learning for Multiagent POMDPs. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, pages 1995–2002, Austin, TX, 2015. AAAI.

J. F. Araujo, P. B. Sujit, and J. B. Sousa. Multiple UAV area decomposition and coverage. In *Proceedings of the 2013 IEEE Symposium on Computational Intelligence for Security and Defense Applications (CISDA)*, pages 30–37, Singapore, 2013. IEEE.

G. Attiya and Y. Hamam. Task allocation for maximizing reliability of distributed systems: A simulated annealing approach. *Journal of Parallel and Distributed Computing*, 66(10):1259–1266, oct 2006.

D. Auger, A. Couëtoux, and O. Teytaud. Continuous Upper Confidence Trees with Polynomial Exploration - Consistency. *Lecture Notes in Computer Science*, 8188 LNAI: 194–209, 2013.

F. Aurenhammer. Voronoi Diagrams-A Survey of a Fundamental Geometric Data Structure. *ACM Computing Surveys (CSUR)*, 23(3):345–405, sep 1991.

I. Baehr. Sharing spatial information for humanitarian response and disaster management. *Geoforum Perspektiv*, 14(26):10–17, 2015.

C. A. B. Baker, S. D. Ramchurn, W. L. Teacy, and N. R. Jennings. Factored Monte-Carlo Tree Search for Coordinating UAVs in Disaster Response. In *ICAPS Proceedings of the 4th Workshop on Distributed and Multi-Agent Planning (DMAP-2016)*, London, 2016a. AAAI.

C. A. B. Baker, S. D. Ramchurn, W. L. Teacy, and N. R. Jennings. Planning Search and Rescue Missions for UAV Teams. In *Conference on Prestigious Applications of Intelligent Systems at ECAI 2016*, pages 1–6, The Hague, The Netherlands, 2016b. ECAI.

J. S. Baras and X. Tan. Control of Autonomous Swarms Using Gibbs Sampling. *Proceedings of the 43rd IEEE Conference on Decision & Control*, 5:4752–4757, 2004.

M. J. Barnes, B. G. Knapp, B. W. Tillman, and B. A. Walters. Crew Systems Analysis of Unmanned Aerial. Technical Report January, Army Research Laboratory, Aberdeen, 2000.

J. Bellingham, M. Tillerson, A. Richards, and J. P. How. *Multi-Task Allocation and Path Planning for Cooperating UAVs*. Doctoral thesis, Massachusetts Institute of Technology, 2001.

Y. Ben-Asher, S. Feldman, P. Gurfil, and M. Feldman. Distributed decision and control for cooperative UAVs Using Ad Hoc communication. *IEEE Transactions on Control Systems Technology*, 16(3):511–516, 2008.

I. E. Ben-Gal. Bayesian Networks. In F. Ruggeri, F. Faltin, and R. Kenett, editors, *Encyclopedia of Statistics in Quality and Reliability*. Wiley, 1st edition, 2007. ISBN 978-0-470-01861-3.

D. M. Benedek, C. Fullerton, and R. J. Ursano. First Responders: Mental Health Consequences of Natural and Human-Made Disasters for Public Health and Public Safety Workers. *Annual review of public health*, 28:55–68, jan 2007.

S. Bernardini, M. Fox, and D. Long. Planning the Behaviour of Low-Cost Quadcopters for Surveillance Missions. In *Proc. of 24th Int. Conference on Automated Planning and Scheduling*, pages 445–453, Portsmouth, NH, 2014. AAAI.

S. Bernardini, M. Fox, and D. Long. Combining temporal planning with probabilistic reasoning for autonomous surveillance missions. *Autonomous Robots*, pages 1–23, 2015.

G. Binetti, D. Naso, and B. Turchiano. Decentralized task allocation for surveillance systems with critical tasks. *Robotics and Autonomous Systems*, 61(12):1653–1664, dec 2013.

M. Bohm and O. Sawodny. Optimal sensor placement for modal based estimation of deformable mirror shape. In *IEEE Conference on Control and Applications (CCA)*, pages 418–423, Sydney, Australia, 2015. IEEE.

G. Brando, D. Rapone, E. Spacone, A. Barbosa, M. Olsen, D. Gillins, R. Soti, H. Varum, A. Arede, N. Vila-Pouca, A. Furtado, J. Oliveira, H. Rodrigues, A. Stavridis, S. Bose, M. Faggella, R. Gigliotti, and R. L. Wood. Reconnaissance report on the 2015 Gorkha earthquake effects in Nepal. In *XVI Convigno Anidis*, L'Aquila, Italy, 2015. Italian National Association of Earthquake Engineering.

G. V. D. Broeck and K. Driessens. Automatic Discretization of Actions and States in Monte-Carlo Tree Search. In *Proceedings of the ECML/PKDD 2011 Workshop on Machine Learning and Data Mining in and around Games*, pages 1–12, 2011.

C. B. Browne, E. J. Powley, D. Whitehouse, S. M. Lucas, P. I. Cowling, P. Rohlfshagen, S. Tavener, D. Perez, S. Samothrakis, and S. Colton. A Survey of Monte Carlo Tree Search Methods. *Transactions on Computational Intelligence and AI in Games*, 4(1): 1–43, 2012.

A. Bry and N. Roy. Rapidly-Exploring Random Belief Trees for Motion Planning Under Uncertainty. In *Proceedings of the 2011 IEEE International Conference on Robotics and Automation*, volume 21, pages 723–730, Shanghai, 2011. Massachusetts Institute of Technology, Cambridge, USA, IEEE.

S. Cameron, S. Hailes, S. Julier, S. McClean, G. Parr, R. Nardi, N. Trigoni, M. Ahmed, G. McPhillips, J. Nie, A. Symington, W. L. Teacy, and S. Waharte. SUAAVE: Combining Aerial Robots and Wireless Networking. In *Proceedings of the 25th Bristol International UAV Systems Conference*, number 01865, pages 1–14, Bristol, 2010. Professional Engineering Publishing.

M. Cap, P. Novak, J. Vokrinek, and M. Pechouvek. Multi-agent RRT: Sampling-based Cooperative Pathfinding. In *Proceedings of the 12th International Conference on Autonomous Agents and Multiagent Systems*, AAMAS '13, pages 1–2, Richland, SC, 2013. IFAAMAS.

O. Cappé, A. Guillin, J. M. Marin, and C. P. Robert. Population Monte Carlo. *Journal of Computational and Graphical Statistics*, 13(4):907–929, 2004.

S. Caselli, M. Reggiani, and R. Rocchi. Heuristic Methods for Randomized Path Planning in Potential Fields. In *Proceedings of the 2001 IEEE International Symposium on Computational Intelligence in Robotics and Automation*, pages 426–431, Alberta, Canada, 2001. IEEE.

J. Casper and R. R. Murphy. Human-Robot Interactions During the Robot-Assisted Urban Search and Rescue Response at the World Trade Center. *IEEE transactions on systems, man, and cybernetics. Part B, Cybernetics : a publication of the IEEE Systems, Man, and Cybernetics Society*, 33(3):367–85, jan 2003.

N. Ceccarelli, M. Pachter, P. Chandler, S. Rasmussen, D. Jacques, C. Schumacher, B. Kish, and T. Shima. *UAV Cooperative Decision and Control*. Society for Industrial and Applied Mathematics, Philadelphia, PN, 1st edition, 2009.

G. M. J.-B. Chaslot, J. W. H. M. Uiterwijk, H. J. V. D. Herik, M. H. M. Winands, and B. Bouzy. Progressive Strategies for Monte-Carlo Tree Search. *New Mathematics and Natural Computation*, 04(03):343–357, 2008.

S. Chen, F. Wu, L. Shen, J. Chen, and S. D. Ramchurn. Multi-Agent Patrolling under Uncertainty and Threats. *PLoS ONE*, 10(6):1–13, 2015.

Y.-b. Chen, G.-c. Luo, Y.-s. Mei, J.-q. Yu, and X.-l. Su. UAV path planning using artificial potential field method updated by optimal control theory. *International Journal of Systems Science*, (October):1–14, jun 2014.

G. Chmaj and H. Selvaraj. UAV Cooperative Data Processing Using Distributed Computing Platform. In *Proceedings of the Twenty-Third International Conference on Systems Engineering*, volume 366, pages 455–461, Las Vegas, NV, 2015. Springer.

Civil Aviation Authority. Unmanned Aircraft: Requirements for operating in airspace, 2016, `http://www.caa.co.uk/unmannedaircraft/` (Accessed: 2016-08-03).

A. Couëtoux. *Monte Carlo Tree Search for Continuous and Stochastic Sequential Decision Making Problems*. Doctoral thesis, Universite Paris Sud, 2013.

A. Couëtoux, J.-B. Hoock, N. Sokolovska, O. Teytaud, and N. Bonnard. Continuous Upper Confidence Trees. In C. A. Coello-Coello, editor, *Proceedings of the 5th International Conference on Learning and Intelligent Optimization*, number 5, pages 433–445, Rome, Italy, 2011a. Springer.

A. Couëtoux, M. Milone, M. Brendel, H. Doghmen, M. Sebag, and O. Teytaud. Continuous Rapid Action Value Estimates. *Journal of Machine Learning Research*, 20:19–31, 2011b.

R. Coulom. Efficient Selectivity and Backup Operators in Monte-Carlo Tree Search. In P. Ciancarni and H. J. V. D. Herik, editors, *5th International Conference on Computer and Games*, volume 4630, pages 72–83, Turin, Italy, 2006.

Crisis Mappers. Crisis Mappers - The Humanitarian Technology Network, 2013, `http://crisismappers.net` (Accessed: 2014-10-30).

M. L. Cummings. Operator Interaction with Centralized Versus Decentralized UAV Architectures. In K. P. Valavanis and G. J. Vachtsevanos, editors, *Handbook of Unmanned Aerial Vehicles*, pages 1–13. Springer, 2013.

M. L. Cummings, S. Bruni, S. Mercier, and P. J. Mitchell. Automation Architecture for Single Operator-Multiple UAV Command and Control. Technical report, NATO, Cambridge, MA, sep 2009.

Da-Jiang Innovations Science and Technology Co. DJI Drones Used To Aid Yunnan Earthquake Relief Efforts, 2014, `http://www.dji.com/newsroom/news/dji-technology-used-to-aid-yunnan-earthquake-relief-efforts` (Accessed: 2016-03-07).

M. de Waard. *Monte Carlo Tree Search with Options for General Video Game Playing*. Masters thesis, Universiteit van Amsterdam, 2016.

F. M. Delle Fave, A. Farinelli, A. Rogers, and N. R. Jennings. A Methodology for Deploying the Max-Sum Algorithm and a Case Study on Unmanned Aerial Vehicles. In *The Twenty-Fourth Innovative Applications of Artificial Intelligence Conference*, pages 2275–2280, Toronto, Canada, 2012a. AAAI.

F. M. Delle Fave, A. Rogers, Z. Xu, S. Sukkarieh, and N. R. Jennings. Deploying the Max-Sum Algorithm for Decentralised Coordination and Task Allocation of Unmanned Aerial Vehicles for Live Aerial Imagery Collection. In *Proceedings of the 2012 IEEE International Conference on Robotics and Automation*, pages 469–476, St Paul, MN, may 2012b. IEEE.

Derbyshire Constabulary. Derbyshire Constabulary - Our Helicopter, 2013, `http://www.derbyshire.police.uk/My-Local-Police/Our-Helicopter/Our-Helicopter.aspx` (Accessed: 2013-06-13).

V. R. Desaraju and J. P. How. Decentralised Path Planning for Multi-Agent Teams in Complex Environments using Rapidly-exploring Random Trees. *2011 IEEE International Conference on Robotics and Automation*, 1(1):4956–4961, 2011.

P. Deville, C. Linard, S. Martin, M. Gilbert, F. R. Stevens, A. E. Gaughan, V. D. Blondel, and A. J. Tatem. Dynamic population mapping using mobile phone data. *Proceedings of the National Academy of Sciences*, 111(45):15888–15893, 2014.

S. R. Dixon, C. D. Wickens, and D. Chang. Mission Control of Multiple Unmanned Aerial Vehicles: A Workload Analysis. *Human Factors: The Journal of the Human Factors and Ergonomics Society*, 47(3):479–487, oct 2005.

K. Durkota and A. Komenda. Deterministic Multiagent Planning Techniques: Experimental Comparison (Short paper). In *Proceedings of DMAP Workshop of ICAPS'13*, pages 43–47, Rome, Italy, 2013. AAAI.

D. Ehrlich, G. Zeug, J. Gallego, A. Gerhardinger, I. Caravaggi, and M. Pesaresi. Quantifying the building stock from optical high-resolution satellite imagery for assessing disaster risk. *Geocarto International*, 25(4):281–293, jul 2010.

J. Ely, C. Graham, I. Barr, B. Rea, M. Spagnolo, and J. Evans. Using UAV acquired photography and structure from motion techniques for studying glacier landforms: application to the glacial flutes at Isfallsglaciären. *Earth Surface Processes and Landforms*, Awaiting P, 2016.

EPSRC. New laser-based aircraft tracking system could aid disaster relief efforts, 2016, `https://www.epsrc.ac.uk/newsevents/news/hyperion/` (Accessed: 2016-07-06).

A. Farinelli, A. Rogers, and N. R. Jennings. Bounded Approximate Decentralised Coordination using the Max-Sum Algorithm. In *IJCAI-09 Workshop on Distributed Constraint Reasoning (DCR)*, pages 46–59, Pasadena, CA, 2010.

A. Farinelli, A. Rogers, and N. R. Jennings. Agent-based decentralised coordination for sensor networks using the max-sum algorithm. *Autonomous Agents and Multi-Agent Systems*, 28(3):337–380, 2014.

A. Farinelli, A. Rogers, A. Petcu, and N. R. Jennings. Decentralised Coordination of Low-Power Embedded Devices Using the Max-Sum Algorithm. In *Proceedings of the 7th International Conference on Autonomous Agents and Multiagent Systems*, pages 12–16. IFAAMAS, 2008.

A. Farinelli, M. Vinyals, A. Rogers, and N. R. Jennings. Distributed Constraint Handling and Optimization. In G. Weiss, editor, *Multiagent Systems*, chapter 12, pages 1–43. MIT Press, Cambridge, MA, 1st edition, 2013.

W. Fawcett and C. S. Oliveira. Casualty Treatment after Earthquake Disasters: Development of a Regional Simulation Model. *Disasters*, 24(3):271–287, 2000.

W. Feller. *An Introduction to Probability Theory and Its Applications. Volume I.* Wiley, 3rd edition, 1968.

E. Feo Flushing, L. M. Gambardella, and G. A. Di Caro. On decentralized coordination for spatial task allocation and scheduling in heterogeneous teams. In *Proceedings of the 15th International Conference on Autonomous Agents and Multiagent Systems*, pages 988–996, Singapore, 2016. IFAAMAS.

J. Fernandez Galarreta, N. Kerle, and M. Gerke. UAV-based urban structural damage assessment using object-based image analysis and semantic reasoning. *Natural Hazards and Earth System Science*, 15(6):1087–1101, 2015.

S. Feyzabadi and S. Carpin. Risk-aware Path Planning Using Hirerachical Constrained Markov Decision Processes. In *Proceedings of the 2014 IEEE International Conference on Automation Science and Engineering*, pages 297–303, Taipei, Taiwan, 2014. IEEE.

J. Fowler. Data curbs earthquake risk in Armenia. Technical report, United Nations Office for Disaster Risk Reduction, 2016, `https://www.unisdr.org/archive/47852`.

Y. Gabriely and E. Rimon. Spiral-STC: An On-Line Coverage Algorithm of Grid Environments by a Mobile Robot. In *Proceedings of the 2002 IEEE International Conference on Robotics and Automation*, volume 1, pages 954–960, Washington, DC, 2002. IEEE.

S. K. Gan, R. Fitch, and S. Sukkarieh. Real-Time Decentralized Search with Inter-Agent Collision Avoidance. In *Proceedings of the 2012 IEEE International Conference on Robotics and Automation*, pages 504–510, St Paul, MN, may 2012. IEEE.

S. Gelly and D. Silver. Monte-Carlo Tree Search and Rapid Action Value Estimation in Computer Go. *Artificial Intelligence*, 175(11):1856–1875, jul 2011.

B. P. Gerkey and M. J. Mataric. A Formal Analysis and Taxonomy of Task Allocation in Multi-Robot Systems. *The International Journal of Robotics Research*, 23(9):939–954, sep 2004.

C. Goerzen, Z. Kong, and B. Mettler. A Survey of Motion Planning Algorithms from the Perspective of Autonomous UAV Guidance. *Journal of Intelligent and Robotic Systems*, 57(1-4):65–100, nov 2009.

A. Goetz, S. Zorn, R. Rose, G. Fischer, and R. Weigel. A Time Difference of Arrival System Architecture for GSM Mobile Phone Localization in Search and Rescue Scenarios. In *Positioning Navigation and Communication (WPNC), 2011 8th Workshop on.*, pages 1–4. IEEE, 2011.

W. L. Goffe, G. D. Ferrier, and J. Rogers. Global optimization of statistical functions with simmulated annealing. *Journal of Econometrics*, 60(1-2):65–99, 1992.

H. Goldingay and J. van Mourik. The effect of load on agent-based algorithms for distributed task allocation. *Information Sciences*, 222:66–80, feb 2013.

Government of the Republic of Haiti. Haiti Earthquake PDNA: Assessment of damage, losses, general and sectoral needs. Technical report, apr 2014.

D. Grosu and A. Das. Auction-based resource allocation protocols in grids. In *Proceedings of the IASTED International Conference on Parallel and Distributed Computing and Systems*, volume 16, pages 20–27, Cambridge, MA, 2004.

C. Guestrin, D. Koller, and R. Parr. Multiagent Planning with Factored MDPs. *Advances in Neural Information Processing Systems*, 14, 2001.

A. Guez, D. Silver, and P. Dayan. Efficient Bayes-Adaptive Reinforcement Learning using Sample-Based Search. *arXiv preprint arXiv:1205.3109*, pages 1–14, 2012.

J. Guitton, J.-L. Farges, and R. Chatila. Cell-RRT: Decomposing the environment for better plan. *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5776–5781, oct 2009.

Harvard Humanitarian Initiative. Disaster relief 2.0: The Future of Information Sharing in Humanitarian Emergencies. Technical report, UN Foundation & Vodafone Foundation Technology Partnership, Washington, D.C. and Berkshire, UK, 2011.

G. Hoffmann and C. Tomlin. Mobile Sensor Network Control Using Mutual Information Methods and Particle Filters. *IEEE TRANSACTIONS ON AUTOMATIC CONTROL, VOL. 55, NO. 1, JANUARY 2010 Mobile*, 55(1):32–47, 2010.

J. Hu, L. Xie, K.-y. K. Y. Lum, and J. Xu. Multiagent Information Fusion and Cooperative Control in Target Search. *IEEE Transactions on Control Systems Technology*, 21(4):1223–1235, jul 2012.

J. Hu, L. Xie, J. Xu, and Z. Xu. Multi-Agent Cooperative Target Search. *Sensors (Switzerland)*, 14(6):9408–9428, 2014.

Y. Huang and K. Gupta. RRT-SLAM for Motion Planning with Motion and Map Uncertainty for Robot Exploration. In *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1077–1082. IEEE, sep 2008.

L. Ingber. Simulated annealing: Practice versus theory. *Mathematical and Computer Modelling*, 18(11):29–57, 1993.

Innovate UK Technology Strategy Board. Competition for funding: Highly innovative technology enablers in aerospace (HITEA) 3. Technical report, Technology Strategy Board, mar 2015.

E. A. Jagla. Minimum energy configurations of repelling particles in two dimensions. *The Journal of Chemical Physics*, 110(1):451, 1999.

L. B. Johnson, H.-l. Choi, and J. P. How. The Role of Information Assumptions in Decentralized Task Allocation. *IEEE Control Systems*, 36(4):45–58, aug 2016.

P. Katsigiannis, L. Misopolinos, V. Liakopoulos, T. K. Alexandridis, and G. Zalidis. An Autonomous Multi-Sensor UAV System for Reduced-Input Precision Agriculture Applications. In *24th Mediterranean Conference on Control and Automation*, pages 60–64, Athens, Greece, 2016. IEEE.

L. Kavraki, L. Kavraki, P. Svestka, P. Svestka, J.-C. Latombe, J.-C. Latombe, M. Overmars, and M. Overmars. Probabilistic Roadmaps for Path Planning in High-Dimensional Configuration Spaces. *Robotics and Automation, IEEE Transactions on*, 12(4):566 – 580, 1996.

M. Kazama and T. Noda. Damage statistics (Summary of the 2011 off the Pacific Coast of Tohoku Earthquake damage). *Soils and Foundations*, 52(5):780–792, 2012.

O. Khatib. The potential field approach and operational space formulation in robot control. In *Proc. of Fourth Yale Workshop on Applicaitons of Adaptive Systems Theory*. Yale University, 1986.

S. Kirkpatrick. Optimization by Simulated Annealing : Quantitative Studies. *Journal of Statistical Physics*, 34(5/6):975–986, 1984.

L. Kocsis and C. Szepesvari. Bandit based Monte-Carlo Planning. *Machine Learning: ECML 2006*, 4212:282–293, 2006.

A. Kolling and A. Kleiner. Multi-UAV Motion Planning for Guaranteed Search. In *Proceedings of the 12th International Conference on Autonomous Agents and Multiagent Systems*, pages 79–86, 2013.

Y. Kong, M. Zhang, and D. Ye. A Negotiation Method for Task Allocation with Time Constraints in Open Grid Environments. *Concurrency and Computation: Practice and Experience*, Early View, apr 2014.

Y. Koren and J. Borenstein. Potential Field Methods and Their Inherent Limitations for Mobile Robot Navigation. In *Proceedings of the 1991 IEEE International Conference on Robotics and Automation*, pages 1398–1404. IEEE, 1991.

M. Kothari and I. Postlethwaite. A Probabilistically Robust Path Planning Algorithm for UAVs Using Rapidly-Exploring Random Trees. *Journal of Intelligent and Robotic Systems*, 71(2):231–253, sep 2012.

M. Kothari, I. Postlethwaite, and D.-W. Gu. Multi-UAV Path Planning in Obstacle Rich Environments Using Rapidly-exploring Random Trees. In *Proceedings of the 48th IEEE Conference on Decision and Control (CDC) held jointly with 2009 28th Chinese Control Conference*, pages 3069–3074, Shanghai, dec 2009. IEEE.

P. Kraaijenbrink, S. W. Meijer, J. M. Shea, F. Pellicciotti, S. M. De Jong, and W. W. Immerzeel. Seasonal surface velocities of a Himalayan glacier derived by automated

correlation of unmanned aerial vehicle imagery. *Annals of Glaciology*, 57(71):103–113, 2016.

F. R. Kschischang, B. J. Frey, and H.-a. Loeliger. Factor Graphs and the Sum-Product Algorithm. *Transacitons on Information Theory*, 47(2):498–519, 2001.

J. Kuffner and S. M. LaValle. RRT-Connect : An Efficient Approach to Single-Query Path Planning. In *2000 IEEE International Conference on Robotics & Automation*, number April, pages 995–1001, San Francisco, CA, 2000. IEEE.

V. Kumar and N. Michael. Opportunities and challenges with autonomous micro aerial vehicles. *The International Journal of Robotics Research*, 31(11):1279–1291, aug 2012.

H. Kurniawati, D. Hsu, and W. Lee. SARSOP: Efficient Point-Based POMDP Planning by Approximating Optimally Reachable Belief Spaces. In *Robotics: Science and Systems*, pages 65–72, 2008.

S. La Cesa, A. Farinelli, L. Iocchi, D. Nardi, M. Sbarigia, and M. Zaratti. Semi-autonomous coordinated exploration in rescue scenarios. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 5001 LNAI:286–293, 2008.

J. P. Laumond, S. Sekhavat, and F. Lamiraux. Introduction and Guidelines in Nonholonomic Motion Planning for Mobile Robots. In J. P. Laumond, editor, *Robot Motion Planning and Control*, volume 229 of *Lecture Notes in Control and Information Sciences*, pages 0–53. Springer-Verlag, London, 1998. ISBN 3-540-76219-1.

S. M. LaValle. Rapidly-Exploring Random Trees: A New Tool for Path Planning. Technical report, Iowa State University, 1998.

S. M. LaValle. *Planning Algorithms*. Cambridge University Press, 2006. ISBN 9788578110796.

T. Lemaire, R. Alami, and S. Lacroix. A Distributed Tasks Allocation Scheme in Multi-UAV Context. In *Proceedings of the 2004 IEEE International Conference on Robotics and Automation*, volume 4, pages 3622–3627. IEEE, 2004.

D. Levine, B. Luders, and J. P. How. Information-rich Path Planning with General Constraints using Rapidly-exploring Random Trees. In *AIAA Infotech@Aerospace Conference*, pages 0–19. AIAA, 2012.

Y. Li, H. Chen, M. Joo Er, and X. Wang. Coverage path planning for UAVs based on enhanced exact cellular decomposition method. *Mechatronics*, 21(5):876–885, 2011.

A. Liekna, E. Lavendelis, and A. Grabovskis. Experimental Analysis of Contract NET Protocol in Multi-Robot Task Allocation. *Applied Computer Systems*, 13(1):6–14, 2012.

F. Y. S. Lin and P. L. Chiu. A Near-Optimal Sensor Placement Algorithm to Achieve Complete Coverage/Discrimination in Sensor Networks. *Communications Letters, IEEE*, 9(1):43–45, 2005.

L. Luo. *Distributed Algorithm Design for Constrained Multi-robot Task Assignment.* Doctoral thesis, Carnegie Mellon University, 2014.

L. J. Luotsinen, A. J. Gonzalez, and L. Boeloeni. Collaborative UAV Exploration of Hostile Environments. In *Proceedings of the 24th, Army Science Conference*, Orlando, FL, 2004. University of Central Florida.

K. S. Macarthur. *Multi-Agent Coordination for Dynamic Decentralised Task Allocation.* Doctoral thesis, Southampton, 2011.

K. S. Macarthur, R. Stranders, S. D. Ramchurn, and N. R. Jennings. A Distributed Anytime Algorithm for Dynamic Task Allocation in Multi-Agent Systems. In *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence*, pages 701–706, San Francisco, CA, 2011. AAAI.

A. G. Macintyre, J. A. Barbera, and B. P. Petinaux. Survival Interval in Earthquake Entrapments: Research Findings Reinforced During the 2010 Haiti Earthquake Response. *Disaster Medicine and Public Health Preparedness*, 5(1):13–22, 2011.

R. Maheswaran, J. Pearce, and M. Tambe. Distributed Algorithms for DCOP: A Graphical-Game-Based Approach. In *Proceedings of the ISCA 17th Conference on Parallel and Distributed Computing Systems*, pages 432–439, San Francisco, CA, 2004. ISCA.

A. Makarenko and H. Durrant-Whyte. Decentralized Bayesian algorithms for active sensor networks. *Information Fusion*, 7(4):418–433, dec 2006.

B. Mathibela, M. A. Osborne, I. Posner, and P. Newman. Can Priors Be Trusted? Learning to Anticipate Roadworks. In *IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC*, pages 927–932. IEEE, 2012.

McKinsey & Company. FDNY Fire Operations response on. Technical report, New York City Fire Department, New York, New York, USA, 2004, `http://www.nyc.gov/html/fdny/pdf/mck_report/fire_operations_response.pdf`.

Measure. Drones for Disaster Response and Relief Operations. Technical report, Washington, D.C., apr 2015.

M. Messinger, G. P. Asner, and M. Silman. Rapid Assessments of Amazon Forest Structure and Biomass Using Small Unmanned Aerial Systems. *Remote Sensing*, 8(615): 1–15, 2016.

N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller. Equation of State Calculations by Fast Computing Machines. *The Journal of Chemical Physics*, 21(6):1087, 1953.

S. M. Mishra, X. Han, D. Sidoti, D. Fernando, M. Ayala, and F. Way. Multi-Objective Coordinated Path Planning for a Team of UAVs in a Dynamic Environment. In *19th International Command and Control Research and Technology Symposium*, Alexandra, VA, 2014. CCRP.

P. J. Modi, W.-m. Shen, M. Tambe, and M. Yokoo. An Asynchronous Complete Method for Distributed Constraint Optimization. In *Proceedings of the 2nd International Conference on Autonomous Agents and Multiagent Systems*, volume 2, pages 161–168, Melbourne, Australia, 2003. IFAAMAS.

N. Morrow, N. Mock, A. Papendieck, and N. Kocmich. Independent Evaluation of the Ushahidi Haiti Project. Technical report, Ushahidi, 2011.

R. R. Murphy. The 100:100 Challenge for Computing in Rescue Robotics. In *2011 IEEE International Symposium on Safety, Security, and Rescue Robotics*, pages 72–75, Kyoto, nov 2011. IEEE.

R. R. Murphy. A Decade of Rescue Robots. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5448–5449, Vilamoura, Portugal, 2012. IEEE.

R. R. Murphy, B. A. Duncan, T. Collins, J. Kendrick, P. Lohman, T. Palmer, and F. Sanborn. Use of a Small Unmanned Aerial System for the SR-530 Mudslide Incident near Oso, Washington. *Journal of Field Robotics*, May:1–13, 2015.

M. Nilges, G. M. Clore, and a. M. Gronenborn. Determination of three-dimensional structures of proteins from interproton distance data by hybrid distance geometry-dynamical simulated annealing calculations. *FEBS letters*, 229(2):317–324, mar 1988.

NOAA. The Devastating Tohoku, Japan, Earthquake and Tsunami of March 11, 2011. Technical report, National Oceanic and Atmospheric Administration (NOAA), 2011.

OCHA. Unmanned Aerial Vehicles in Humanitarian Response. Technical Report June, UN Office for Coordination of Humanitarian Affairs: Policy Development and Studies Branch, 2014.

A. Ollero, J. Martínez-de Dios, and L. Merino. Unmanned Aerial Vehicles as tools for forest-fire fighting. *Forest Ecology and Management*, 234(1):S263, nov 2006.

Orchid Project. Disaster response, 2014, `http://www.orchid.ac.uk/disaster-response-2/` (Accessed: 2016-07-06).

M. W. Otte. *Any-Com Multi-Robot Path Planning*. Phd thesis, University of Colorado, Bolder, 2011.

D. Pascarella, S. Venticinque, R. Aversa, M. Mattei, and L. Blasi. Parallel and distributed computing for UAVs trajectory planning. *Journal of Ambient Intelligence and Humanized Computing*, 6(6):773–782, 2015.

N. N. Patel, F. R. Stevens, Z. Huang, A. E. Gaughan, I. Elyazar, and A. J. Tatem. Improving Large Area Population Mapping Using Geotweet Densities. *Transactions in GIS*, 20(3), 2016.

T. Penya-Alba and M. Vinyals. A Scalable Message-Passing Algorithm for Supply Chain Formation. In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence*, number June, pages 1436–1442, Toronto, Canada, 2012. AAAI.

A. Petcu and B. Faltings. MB-DPOP : A New Memory-Bounded Algorithm for Distributed Optimization. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1452–1457. IJCAI, 2007.

J. Pineau and G. G. Gordon. POMDP Planning for Robust Robot Control. In *The Twelveth International Symposium on Robotics Research*, pages 1–10, San Francisco, CA, 2005. Springer.

Police of Hungary. Simulation reproducing the industrial spill at the Ajka Alumina Plant, 2010, `http://index.hu/belfold/2010/10/08/modelleztek_a_tobbi_zagytarozot/` (Accessed: 2016-08-03).

E. J. Powley, D. Whitehouse, and P. I. Cowling. Monte Carlo Tree Search with macro-actions and heuristic route planning for the Physical Travelling Salesman Problem. In *Proceedings of the 2012 IEEE Conference on Computational Intelligence and Games*, pages 234–241, Granada, Spain, 2012. IEEE.

PrecisionHawk. INTRODUCING THE LANCASTER 5, 2016, `http://precisionhawk.com/lancaster` (Accessed: 2016-07-27).

M. Pujol-Gonzalez, J. Cerquides, A. Farinelli, P. Meseguer, and J. A. Rodriguez-Aguilar. Efficient Inter-Team Task Allocation in RoboCup Rescue. In *Proceedings of the 14th International Conference on Autonomous Agents and Multiagent Systems*, pages 413–421, Istanbul, Turkey, 2015. IFAAMAS.

M. Raissi-Dehkordi, K. Chandrashekar, and J. S. Baras. UAV Placement for Enhanced Connectivity in Wireless Ad-hoc Networks. Technical report, Center for Satellite and Hybrid Communication Networks, 2004.

S. D. Ramchurn, A. Farinelli, K. S. Macarthur, and N. R. Jennings. Decentralized Coordination in RoboCup Rescue. *The Computer Journal*, 53(9):1447–1461, mar 2010.

S. D. Ramchurn, J. E. Fischer, Y. Ikuno, F. Wu, J. Flann, and A. Waldock. A Study of Human-Agent Collaboration for Multi-UAV Task Allocation in Dynamic Environments. In *Proceedings of the 24th International Joint Conference on Articificial Intelligence (IJCAI)*, pages 1184–1192, Buenos Aires, Argentina, 2015. IJCAI.

P.-y. Richard. A Task Allocation Model for Distributed Computing Systems. *IEEE Transactions on Computers*, C-31(1):41–47, jan 1982.

A. Rogers, A. Farinelli, R. Stranders, and N. R. Jennings. Bounded approximate decentralised coordination via the max-sum algorithm. *Artificial Intelligence*, 175(2): 730–759, 2011.

P. Rolet, M. Sebag, and O. Teytaud. Upper Confidence Trees and Billiards for Optimal Active Learning. In *Proc. Conf. l'Apprentissage Autom. (CAP09)*, Hammamet, Tunisia, 2009. HAL.

S. Salapaka, A. Khalak, and M. A. Dahleh. Constraints on locational optimization problems. In *Proceedings of the 42nd IEEE Conference on Decision & Control*, number December, pages 1741–1746, Maui, HI, 2003. IEEE.

R. Sawhney, K. M. Krishna, and K. Srinathan. On Fast Exploration in 2D and 3D Terrains with Multiple Robots. In Decker, Sichman, Sierra, and Castelfranchi, editors, *Proceedings of the 8th International Conference on Autonomous Agents and Multiagent Systems*, pages 73–80, Budapest, Hungary, 2009. IFAAMAS.

P. Scerri, A. Farinelli, S. Okamoto, and M. Tambe. Allocating Tasks in Extreme Teams. In *Proceedings of the 4th International Conference on Autonomous Agents and Multiagent Systems*, page 727, New York City, NY, 2005. IFAAMAS.

P. Scerri, T. V. Gonten, and G. Fudge. Transitioning Multiagent Technology to UAV Applications. In Berger, Burg, and Nishiyama, editors, *Proceedings of the 7th International Conference on Autonomous Agents and Multiagent systems*, Estoril, Portugal, 2008. IFAAMAS.

P. Scerri, Y. Xu, E. Liao, J. Lai, and K. Sycara. Scaling Teamwork to Very Large Teams. In *Proceedings of the 3rd International Conference on Autonomous Agents and Multiagent Systems*, pages 888–895. IEEE, 2004.

G. Settembre, P. Scerri, A. Farinelli, K. Sycara, and D. Nardi. A decentralized approach to cooperative situation assessment in multi-robot systems. In *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems*, pages 31–38, 2008.

W. M. Shen and B. Salemi. Distributed and Dynamic Task Reallocations in Robot Organization. In *Proceedings of the 2002 IEEE International Conference on Robotics and Automation*, pages 1019–1024, Washington, DC, 2002. IEEE.

A. Singh, A. Krause, and W. Kaiser. Nonmyopic Adaptive Informative Path Planning for Multiple Robots. In *Proceedings of the Twenty-First International Joint Conference on Artificial Intelligence*, pages 1843–1850. Center for Embedded Network Sensing (CENS), IJCAI, 2009.

R. Smallwood and E. Sondik. The Optimal Control of Partially Observable Markov Processes Over a Finite Horizon. *Operations Research*, 21(5):1071–1088, 1973.

R. G. Smith. The Contract Net Protocol: High-Level Communication and Control in a Distributed Problem Solver. *IEEE Transactions on Computers*, C-29(12):1104–1113, 1980.

E. Sommerlade and I. Reid. Probabilistic surveillance with multiple active cameras. In *2010 IEEE International Conference on Robotics and Automation*, pages 440–445. IEEE, may 2010.

M. T. J. Spaan, F. A. Oliehoek, and C. Amato. Scaling Up Optimal Heuristic Search in Dec-POMDPs via Incremental Expansion. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI)*, Barcelona, Spain, 2011. IJCAI.

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

P. B. Sujit and R. Beard. Multiple UAV exploration of an unknown region. *Annals of Mathematics and Artificial Intelligence*, 52(2-4):335–366, mar 2009.

P. B. Sujit and S. Saripalli. Unmanned Aerial Vehicle Path Following. *IEEE Control Systems Magazine*, (February):42–59, feb 2014.

P. B. Sujit, A. Sinha, and D. Ghose. Multiple UAV task allocation using negotiation. In *Proceedings of the 5th International Conference on Autonomous Agents and Multiagent Systems*, page 471, New York, NY, 2006. ACM Press.

R. S. Sutton and A. G. Barto. *Reinforcement Learning, An Introduction*. MIT Press, Cambridge, MA, 3rd edition, 2000. ISBN 0-262-19398-1.

S. Tadokoro. Earthquake Disaster and Expectation for Robotics. In S. Tadokoro, editor, *Rescue Robotics*, chapter 1, pages 1–16. Springer, London, 2009.

The IAFC Board of Directors. IAFC Position: Use of Unmanned Aerial Vehicles in Public Safety Emergency Response, 2014, `http://www.iafc.org/Admin/ResourceDetail.cfm?ItemNumber=7356` (Accessed: 2015-01-15).

J. Tisdale, Z. W. Kim, and J. K. Hedrick. Autonomous UAV path planning and estimation: An online path planning framework for cooperative search and localization. *IEEE Robotics and Automation Magazine*, 16(2):35–42, 2009.

UNICEF. Reaching the Unreached. Nepal Earthquake: Six Months Review. Technical report, United Nations Children's Fund (UNICEF) Nepal Country Office, 2015.

United States Government Accountability Office. Assessments of Selected Weapons Programs. Technical report, Government of the United States, 2013.

US Air Force. RQ-4 Global Hawk, 2014, http://www.af.mil/AboutUs/FactSheets/Display/tabid/224/Article/104516/rq-4-global-hawk.aspx (Accessed: 2016-07-27).

US-Government. Broad Agency Announcement DARPA Robotics Challenge Tactical Technology Office (TTO), 2012, http://www.androidworld.com/DARPA_Robotics_Challenge.pdf.

USAID. Nepal Earthquake Fact Sheet. Technical report, United States Agency for International Development, dec 2015.

J. Valasek, J. V. Henrickson, E. Bowden, Y. Shi, C. L. S. Morgan, and H. L. Neely. Multispectral and DSLR sensors for assessing crop stress in corn and cotton using fixed-wing unmanned air systems. *Autonomous Air and Ground Sensing Systems for Agricultural Optimization and Phenotyping*, 9866(98660L), 2016.

P. G. van der Velden, P. van Loon, C. C. Benight, and T. Eckhardt. Mental health problems among search and rescue workers deployed in the Haïti earthquake 2010: a pre-post comparison. *Psychiatry research*, 198(1):100–5, jun 2012.

M. Venanzi. *Trust-Based Algorithms for Fusing Crowdsourced Estimates of Continuous Quantities*. Doctoral thesis, University of Southampton, 2014.

M. Venanzi, W. L. Teacy, A. Rogers, and N. R. Jennings. Bayesian Modelling of Community-Based Multidimensional Trust in Participatory Sensing under Data Sparsity. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence (IJCAI)*, volume January, pages 717–724. IJCAI, 2015.

VT-Group. Devastating Earthquake in Haiti Pioneering Technology Application Saves Time & Resources, 2011, http://www.vt-group.com/wordpress/wp-content/uploads/2012/11/HaitiEarthquakeUnmannedServices.pdf.

S. Waharte and N. Trigoni. Supporting Search and Rescue Operations with UAVs. In *Proceedings of the 2010 International Conference on Emerging*, pages 142–147. IEEE, 2010.

S. Waharte, N. Trigoni, and S. Julier. Coordinated Search with a Swarm of UAVs. *2009 6th IEEE Annual Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks Workshops*, pages 1–3, jun 2009.

L. Wallace, A. Lucieer, Z. Malenovsk???, D. Turner, and P. Vop??nka. Assessment of forest structure using two UAV techniques: A comparison of airborne laser scanning and structure from motion (SfM) point clouds. *Forests*, 7(3):1–16, 2016.

Y. Wang, J.-Y. Audibert, and R. Munos. Algorithms for Infinitely Many-Armed Bandits. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, *Advances*

in Neural Information Processing Systems, pages 1729–1736. NIPS, 21 edition, 2008. ISBN 9781605609492.

A. Weinstein and M. Littman. Bandit-Based Planning and Learning in Continuous-Action Markov Decision Processes. In *Proc. of 22nd Int. Conference on Automated Planning and Scheduling*, pages 306–314, Sao Paulo, Brazil, 2012. AAAI.

L. T. Wille and J. Vennik. Computational complexity of the ground-state determination of atomic clusters. *Journal of Physics A: Mathematical and Theoretical*, 18(8):L419, 1985a.

L. T. Wille and J. Vennik. Electrostatic energy minimisation by simulated annealing. *Journal of Physics A: Mathematical and Theoretical*, 1113(17):L1113, 1985b.

R. Wilson, E. zu Erbach-Schoenberg, M. Albert, D. Power, S. Tudge, M. Gonzalez, S. Guthrie, H. Chamberlain, C. Brooks, C. Hughes, L. Pitonakova, C. Buckee, X. Lu, E. Wetter, A. Tatem, and L. Bengtsson. Rapid and Near Real Â Time Assessments of Population Displacement Using Mobile Phone Data Following Disasters : The 2015 Nepal Earthquake. *PLoS Currents*, (1):1–26, 2016.

S. Wong and B. MacDonald. A topological coverage algorithm for mobile robots. In *Proceedings of the 2003 IEEE/RSJ Intl. Conference on Intelligent Robots and Systems (IROS)*, number October, pages 1685–1690, Las Vegas, NV, 2003. IEEE.

M. Wooldridge. *An Introduction To Multiagent Systems*. Wiley, Chichester, 2008. ISBN 978-0-471-49691-5.

M. Wooldridge and N. R. Jennings. Intelligent Agents: Theory and practice. *The Knowledge Engineering Review*, 10(2):115–152, 1995.

W. Xinggang, G. Cong, and L. Yibo. Variable Probability based Bidirectional RRT Algorithm for UAV Path Planning. In *The 26th Chinese Control and Decisions Conference*, number 6, pages 2217–2222, Changsha, China, 2014. IEEE.

F. Yamazaki, W. Liu, R. Bahri, and T. Sasagawa. Damage extraction of buildings in the 2015 Gorkha, Nepal earthquake from high-resolution SAR data. In *Land Surface and Cryosphere Remote Sensing III*, volume 9877, pages 98772K1–11, New Delhi, India, 2016. SPIE.

F. Yamazaki, T. Matsuda, S. Denda, and W. Liu. Construction of 3D models of buildings damaged by earthquakes using UAV aerial images. In *Proceedings of the Tenth Pacific Conference on Earthquake Engineering Building an Earthquake-Resilient Pacific*, Sydney, Australia, 2015.

K. Yang, S. Keat Gan, and S. Sukkarieh. A Gaussian process-based RRT planner for the exploration of an unknown and cluttered environment with a UAV. *Advanced Robotics*, (February):1–13, feb 2013.

Y. Yang, A. Minai, and M. Polycarpou. Decentralized Cooperative Search by Networked UAVs in an Uncertain Environment. In *Proceedings of the 2004 American Control Conference*, pages 5558–5563, Boston, MA, 2004. AACC.

H. Yedidsion, R. Zivan, and A. Farinelli. Explorative Max-sum for Teams of Mobile Sensing Agents. In *Proceedings of the 12th International Conference on Autonomous Agents and Multiagent Systems*, pages 549–556, Paris, France, 2014. IFAAMAS.

M. Zaera, M. Esteve, J. C. Guerri, F. J. Martinez, and C. D. Vera. Real-Time Scheduling and Guidance of Mobile Robots on Factory Floors Using Monte Carlo methods Under Windows NT. In *Proceedings of the 8th International IEEE Conference on Emerging Technologies and Factory Automation*, pages 67–74, Antibes - Juan les Pins, France, 2001. IEEE.

R. R. Zargar, M. Sohrabi, M. Afsharchi, and S. Amani. Decentralized Area Patrolling for teams of UAVs. In *4th International Conference on Control, Instrumentation, and Automation (ICCIA)*, pages 27–28, Qazvin, Iran, 2016. Qazvin Islamic Azad University, IEEE.

Z. Zhen, C. Gao, Q. Zhao, and R. Ding. Cooperative Path Planning for Multiple UAVs Formation. In *The 4th Annual IEEE International Conference on Cyber Technology in Automation, Control and Intelligent Systems*, volume 210016, pages 469–473, Hong Kong, China, 2014. IEEE.

S. Zorn, R. Rose, A. Goetz, and R. Weigel. A Novel Technique for Mobile Phone Localization for Search and Rescue Applications. In *2010 International Conference on Indoor Positioning and Indoor Navigation*, number September, pages 15–17, Zurich, Switzerland, 2010. IEEE.