UNIVERSITY OF SOUTHAMPTON

FACULTY OF PHYSICAL SCIENCES AND ENGINEERING

Electronics and Computer Science

Collaborative Search and Rescue by Autonomous Robots

by

Zoltán Beck

Thesis for the degree of Doctor of Philosophy

December 2016

UNIVERSITY OF SOUTHAMPTON

ABSTRACT

FACULTY OF PHYSICAL SCIENCES AND ENGINEERING
Electronics and Computer Science

Doctor of Philosophy

COLLABORATIVE SEARCH AND RESCUE BY AUTONOMOUS ROBOTS

by Zoltán Beck

In recent years, professional first responders have started to use novel technologies at the scene of disasters in order to save more lives. Increasingly, they use robots to search disaster sites. One of the most widely and successfully used robot platforms in the disaster response domain are *unmanned aerial vehicles* (UAVs). UAVs allow remote inspection and mapping. They are able to provide high resolution imagery and often need minimal infrastructure to fly. This allows settings where multiple UAVs are airborne accelerating the information gathering from the disaster site. However, current deployments use labour intensive, individually teleoperated UAVs. Given this, there is a drive toward using multiple robots operating with a certain level of autonomy, in order to decrease the operators' workload. One approach for utilising multiple robots in this way is semi-autonomous operation supervised by a small number of professionals; only requiring human experts for crucial decisions. Current commercial UAV platforms also allow the deployment of a diverse group of robots, allowing them to combine their individual capabilities to be more efficient. For example, *fixed-wing* UAVs are capable of flying faster and carry larger payload, but when they do so, they should be deployed with higher safety measures (safety pilots are required for non-lightweight aircraft). On the other hand, small *rotary-wing* UAVs are more agile and can approach and provide imagery about objects on the ground.

To this end, this thesis develops a number of new approaches for the collaboration of a heterogeneous group of robots in disaster response. More specifically, the problem of collaborative planning with robots operating in an uncertain workflow based setting is investigated by solving the search and rescue (SAR) collaboration problem. Of course, the problem complexity increases when collaborating with different robots. It is not different in this setting, the actions of different types of robots need to be planned with dependencies between their actions under uncertainty.

To date, research on collaboration between multiple robots has typically focused on known settings, where the possible robot actions are defined as a set of tasks. However, in most real-world settings, there is a significant amount of uncertainty present. For example, information about a disaster site develops gradually during disaster relief, thus initially there is often very little certainty about the locations of people requiring assistance (e.g. damaged buildings, trapped victims, or supply shortages). Existing solutions that tackle collaboration in the face of uncertain information are typically limited to simple exploration or target search problems. Moreover, the use of generic temporal planners rapidly becomes intractable for such problems unless applied in a domain-specific manner. Finally, domain specific approaches rarely involve complex action relations, such as task dependencies where the actions of some robots are built on the actions of others. When they do so, decomposition techniques are applied to decrease the problem complexity, or simple heuristics are applied to enhance similar collaboration. Such approaches often lead to low quality solutions, because vital action dependencies across different roles are not taken into account during the optimisation.

Against this background, we offer novel online planning approaches for heterogeneous multi-robot collaboration under uncertainty. First, we provide a negotiation-based bidirectional collaborative planning approach that exploits the potential in determinisation via hindsight optimisation (HOP) combined with long-term planning. Second, we extend this approach to create an anytime Monte Carlo tree search planner that also utilises HOP combined with long-term planning. In online planning settings, such as SAR, anytime planners are beneficial to ensure the ability of providing a feasible plan within the given computational budget. Third, we construct a scenario close to physical deployment that allows us to show how our long-term collaborative planning outperforms the current state of the art path-planning approaches by 25 %.

We conclude that long-term collaborative planning under uncertainty provides an improvement when planning in SAR settings. When combined, the contributions presented in this thesis represent an advancement in the state of the art in the field of online planning under uncertainty. The approaches and methods presented can be applied in collaborative settings when uncertainty plays an important role for defining dependencies between partial planning problems.

Contents

Declaration of Authorship

I, Zoltán Beck, declare that the thesis entitled *Collaborative Search and Rescue by Autonomous Robots* and the work presented in the thesis are both my own, and have been generated by me as the result of my own original research. I confirm that:

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

Signed:..

Date:...

Acronyms

AUV	autonomous underwater vehicle
CNP	contract net protocol
DCOP	distributed constraint optimisation
DEC-MDP	decentralised Markov decision process
DOF	degrees of freedom
DST	Dempster-Shafer theory
GDL	general distributive law
GSM	Global System for Mobile Communications
HOP	hindsight optimisation
ICCID	integrated circuit card identifier
KL	Kullback-Leibler
LCB	lower confidence bound
MCTS	Monte Carlo tree search
MDP	Markov decision process
MGM	maximum gain message
MILP	mixed integer linear programming
MRTA	multi-robot task allocation
OCHA	United Nations Office for the Coordination of Humanitarian Affairs
POMDP	partially observable Markov decision process
RF	radio frequency
RRT	rapidly exploring random tree
SAPT	shortest adjusted processing time first
SAR	search and rescue
TD	to dominant
TE	to equal
TN	to non-dominant
UAV	unmanned aerial vehicle
UCB	upper confidence bound
UGV	unmanned ground vehicle
UMRTA	uncertain multi-robot task allocation
UMV	unmanned maritime vehicle
UN	United Nations

UNITAR	United Nations Institute for Training and Research
UNOSAT	UNITAR Operational Satellite Applications Programme
USCG	United States Coast Guard
VoI	value of information

Nomenclature

$U(\tau)$	Utility resulting from performing task τ.
U_0	Initial task utility.
γ	Task utility decrease factor.
$t(\tau), t(\tau, s)$	Time of competition of task τ (within schedule s).
$\mathbf{T}, \mathbf{T}_s$	Set of tasks and rescue tasks.
$\mathbb{T}_r$	Set of all possible rescue tasks.
$\mathbf{T}_r, \mathcal{T}_r^+$	Set of known and random set of unknown rescue tasks.
$\mathbf{R}, \mathbf{R}_s, \mathbf{R}_r$	Set of robots, search robots and rescue robots respectively.
$Poisson(\lambda)$	2D inhomogeneous spatial Poisson point process with intensity function λ.
$\mathbf{D} = [0, 2\pi)$	Set of possible motion directions.
$AP\,(\tau, s)$	Adjusted processing time of task τ after schedule s.
$[s, \tau]$	Schedule where task τ is inserted at the end of schedule s.
$trav(\tau_i, \tau_j)$	Necessary travel time between task τ_i and τ_j.
$P(\tau)$	Process time of task τ.
$dir(R, \tau)$	Direction that robot R has to take to move towards task τ.
$w(s)$	Weight of a schedule in the aggregation process, representing the number of delayed tasks if the first task is delayed.
$cstr(\tau)$	Temporal constraint on the execution of task τ from the rescue plans.
$D\,(s_s)$	Overall delay on the rescue tasks caused by search schedule s_s.
$\Delta D\,(s_s, \tau, i)$	Increase in the delay on the rescue tasks when inserting task τ in search schedule s_s at position i.
ϵ	Percentile ratio.
$U_{b,\varepsilon}$	Percentile lower boundary utility.
U_{max}	Maximum utility of children.
$\hat{U}_\varepsilon$	Percentile mean utility.
$\phi(x), \phi(x\|\mu, \sigma^2)$	Probability density function of a normal distribution with μ (or 0) mean and σ^2 (or 1) variance.
$\Phi(x), \Phi(x\|\mu, \sigma^2)$	Cumulative distribution function of a normal distribution with μ (or 0) mean and σ^2 (or 1) variance.
$D(\alpha)$	Schedule delay due to α direction error of the motion of a rescue robot.

$\overline{D(\alpha)}$	Mean schedule delay due to α direction error of the motion of a rescue robot.
$\Delta U_R(\alpha)$	Utility decrease due to α direction error of the motion of a rescue robot.
$\overline{\Delta U_R(\alpha)}$	Mean utility decrease due to α direction error of the motion of a rescue robot.
$\omega(d)$	Weight function for the weighted random choice for finding random solutions for the MRTA problem of a task with d adjusted processing time difference.
$m_{x,k}^t$	GSM signal measurement.
x	The possible location of a mobile phone.
$\mathbf{l}_k^t$	The location of a GSM receiver.
$d_{x,k}^t$	Distance estimate between the mobile phone and the receiver location.
m_0	Mean GSM signal measurement at reference distance.
d_0	Reference distance for mean GSM signal measurement.
n	Path loss exponent for GSM signals.
χ	Noise component in the GSM signal measurement.
$\overline{d}_{x,k}$	Expected distance for a specific GSM signal measurement.
$\hat{m}$	GSM sensing threshold for detecting a GSM device.
$\mathbf{D} = \{m_{x,k}^t\}$	Set of GSM measurements for all $t > 0$, and $k \in \mathbf{R}_s$.
$\mathbf{D}_x$	Set of GSM measurements until when the GSM device x is first detected ($t_r > t > 0$).
$P(x\|D)$	Posterior of a GSM device's location.
$P_0(x)$	Initial posterior based on the information about the disaster's impact.
$\lambda(x\|D)$	The intensity function of the distribution of unobserved GSM devices.
$\lambda_0(x)$	The initial intensity function of the distribution of unobserved GSM devices based on the information about the disaster's impact.
v_s	Visual victim search speed of a rotary-wing UAV $\left[\frac{\mathrm{m}^2}{s}\right]$.
ΔA_τ	Decrease in the visual victim search area of task τ.
$\Delta U_\tau^+(\Delta t)$	Utility increase by finishing task τ Δt earlier.
$\Delta U_s^-(\delta t, \Delta t)$	Utility decrease caused by increasing the delay of schedule s from Δt to $\Delta t + \delta t$.

Chapter 1

Introduction

First response is the first and often most critical phase of disaster relief. These first few worst days after a disaster hits an area are crucial for saving lives and damaged structures, and decreasing further impact. As technology advances, it is becoming more common that robots aid the work of first responders. Currently a specific type of robot platforms, unmanned aerial vehicles (UAVs), dominate in the disaster response domain (Leetaru, 2015) after numerous platforms recently became available in the market. UAVs allow remote inspection and mapping. They are able to provide high resolution imagery and often need minimal infrastructure to fly. This allows settings where multiple UAVs are airborne accelerating the information gathering from the disaster site. However, current deployments use labour intensive, individually teleoperated robots or UAVs. Given this, there is a drive toward using multiple robots operating with a certain level of autonomy, in order to decrease the operators' workload. One approach for utilising multiple robots in this way is semi-autonomous operation supervised by a small number of professionals; only requiring human experts for crucial decisions (Ramchurn et al., 2015). Current commercial platforms also allow the deployment of a diverse group of robots, allowing them to combine their individual capabilities to be more efficient (Brutschy et al., 2012). This brings the need for these robots to collaborate in complex, real-life settings to support tasks such as search and rescue (SAR).

Against this background, in this chapter we discuss how robots can aid disaster relief, especially focusing on SAR where it is necessary to cope with the rapid changes in the available information. In more detail, we introduce SAR, then introduce current applications of robotics in SAR. After that, we formulate the key problem of this thesis for using robotics for SAR. This is followed by our research contributions in the field of multi-robot online planning in complex and uncertain collaborative settings. Finally, we give an outline of the rest of the thesis.

1.1 Search and Rescue

SAR is the search for and provision of aid to people in distress or imminent danger such as in a disaster setting (USCG, 1995). It involves the collection of information about victims, as well as making efforts to aid them. This combined process is executed by a team of professionals, known as first responders. In many cases, such teams are organised into a three-level hierarchical structure (Punch and Markham, 2000).

- *Gold command*: A small group of high level coordinators responsible for logistics and general resource and information management during first response. They might operate from a remote headquarters.
- *Silver command*: Low level coordinators dealing with the organisation of the SAR teams and other participants operating at the disaster site. Usually based in local headquarters.
- *Bronze command*: First responder professionals conducting the search and rescue operations, find and aid victims at the disaster site.

There are existing software tools for resource allocation and high level coordination that the gold command can utilise (Tressel et al., 2014), while silver command can also make use of several planning software solutions (Ramchurn et al., 2016). On the other hand, there are very few tools available for bronze commanders. The target of this work is to find ways to utilise UAVs for information collection to help the work of the bronze command.

The most typical case for information collection with UAVs is through aerial imagery. As discussed in United Nations Foundation et al. (2010), aerial imagery can be critical for making operational decisions in disaster relief. Aerial imagery of a disaster site is essential to estimate the impact in different areas (UNOSAT, 2011; Pesaresi et al., 2010) and to have a general overview of the area for first responders in terms of logistics or accessibility of areas. Unfortunately, as the following examples show, it is often collected too late for the first wave of disaster response.

- On 12th January 2010 an earthquake hit Haiti with a magnitude of 7.0 causing over 200,000 deaths. SAR teams immediately started to look for survivors under the thousands of collapsed buildings in the capital (Port-au-Prince) and the surrounding area. However, only half of the SAR sectors were searched by first responders a week after the disaster in the southern area of Port-au-Prince (MapAction, 2010). Aerial imagery collection did not allow first responders to have a global picture of the disaster in the first days, as satellite images were available after one day, and high resolution aerial imagery was only available 6 days after the earthquake.

- On 4th October 2010 at the Ajka alumina plant (Ajka, Hungary) one of the largest industrial spills in Europe contaminated an area of 15 square miles, including three nearby villages, with strong alkaline red mud (pH 13). The disaster caused the death of 10 and injured over 150 people. A similar delay can be seen in aerial imagery collection, as it was available only after two days. In the first few days, the still flowing, meter deep red mud made searching for trapped persons difficult until aerial imagery was available (Guy Carpenter, 2010).

As detailed in Pesaresi et al. (2010), aerial imagery conveys crucial information, and can be the basis of estimating several important parameters of the disaster either using computer vision or human visual inspection. These include estimates of the:

- numbers of people in refugee camps,
- characterisation of slum or dwellers,
- assessment of destroyed/rebuilt buildings,
- estimation of built-up stock in cities,
- monitoring refugee camp decommissioning.

Besides these features, recent or near real-time imagery can be used for identifying individual hazards or possible victims (Andriluka et al., 2010). Imagery collected by UAVs can save lives by identifying critical locations via rapid damage assessment, or directly assisting SAR teams (OCHA, 2014). In all of these cases, it is crucial that such information is collected in a *time-critical* manner, as there is a rapid decrease in the ratio for successful rescues in the first few days after a disaster (Coburn et al., 1991; Kawata, 1995; Marconi et al., 2013).

Another trend in disaster response is communication with the locals at a disaster site (Grath, 2014). In many cases, the authorities and aid organisations try to make use of the mobile phones of the victims to gather information of specific needs and incidents (Morrow et al., 2011). As most people can access a mobile phone nowadays, even in developing regions, using these devices to make contact with and find victims at a disaster can be a powerful tool. For these reasons recent studies and experiments show the possibility of locating people in distress using mobile phones (Pastor-Escuredo et al., 2014; Nakanishi, 2016). Given this, Chapter 5 describes an automated victim search system utilising the mobile phones of the victims.

Besides aerial imagery and data from mobile phones, there are many other types of information available during first response. For example, the United Nations Foundation et al. (2010) introduces the complex information flow structure of disaster response while pointing out lessons learned from the Haiti earthquake relief. After a disaster happens, there is very little known about where and how to help victims. Available information gradually increases as various types of information are gathered during disaster relief. This means incoming information might drastically change the required effort from first responders from one moment to another. For example, information could come about high priority rescue tasks that change the planned route of bronze commanders or satellite images can be processed providing information about impacted areas, restricting victim search to these areas only. This means that a system aiming to be deployed for first response purposes has to be *flexible* enough to keep up with incoming information changing the knowledge about the disaster site.

Drawing this all together, the efficiency of SAR in the first few days after a disaster is crucial in terms of saving lives. Latest advances in robotics, more specifically UAV technologies, could provide a very efficient tool for accelerating the work of SAR teams. With this in mind, the application of robotics in a disaster settings will be elaborated in the following section.

1.2 Robots in Disaster Response

There are several advantages of using robots in SAR. They can be used to access areas that are difficult or dangerous to approach for first responders. Examples for this is assessing a damaged building (Angermann et al., 2012) or an injured nuclear reactor (Nagatani et al., 2013). Furthermore, UAVs are able to travel quickly over the disaster site, where ground transport is almost always restricted due to road blockages. These abilities can be very useful for rapid assessment or package (food, medication) delivery (OCHA, 2014). Another advantage is that smaller platforms are much easier to transfer to the disaster site in larger quantities than disaster response professionals. When multiple robots operate on the disaster site, the time to cover a specific area significantly reduces.

However, current deployments use labour intensive, individually teleoperated UAVs (Lubrano, 2013; Goodrich et al., 2008). In recent years, several UAV platforms have been introduced with some degree of autonomy, providing alternatives to this practice. Specifically, autonomy in this context means that the system is able to make decisions without explicit human commands. In terms of research, there is an ongoing focus on utilising a system of multiple autonomous UAVs (Ramchurn et al., 2015; Han et al., 2013; Scerri et al., 2007). These solutions utilise a homogeneous group of UAVs, while current commercial UAV platforms allow the deployment of a diverse group of robots, allowing them to combine their individual capabilities to be more efficient (Brutschy et al., 2012). According to this, this work focuses on utilising a *heterogeneous* group of autonomous UAVs for disaster response.

When robots operate with a certain level of autonomy, they can be considered as agents in a *multiagent* system. Agents are defined as interacting autonomous actors perceiving and making changes in their environment (Wooldridge and Jennings, 1995). In disaster response, a robot, a computer system, a piece of software or a first responder could be considered as an agent, so their relations or collective behaviour can be modeled as a multiagent system. There are many examples of a group of collaborative robots to be presented as a multiagent system (Fiedrich and Burghardt, 2007; Scerri et al., 2008; Belbachir et al., 2010; Gerkey and Mataric, 2002). In this context, the behaviour of the autonomous vehicles is broken down into decisions of individual agents in a multiagent system. This allows critical decisions to be made locally by agents that enables rapid reaction to information changes, increasing the flexibility of the system, and addressing the problem outlined in Section 1.1.

In particular, there are three types of robots applied in rescue robotics (Tadokoro, 2009):

- Unmanned ground vehicles (UGVs) can be used to search areas that are dangerous or difficult to access,
- Unmanned aerial vehicles (UAVs) are capable of remote sensing by providing aerial imagery or live video feed from distant locations,
- Unmanned maritime vehicles (UMVs) can traverse on or under the water surface that can be helpful over the sea or in case of flooding.

There is a high interest in deploying UGVs in disaster response and many competitions encourage research in this field. For example, the RoboCupRescue Robot League promotes research in robotics systems for indoors exploration and rescue (Akin et al., 2013), while the DARPA Robotics Challenge aims to involve humanoid robots in disaster response (Levi et al., 2013). Furthermore, new types of agile UGV platforms are currently being developed for searching victims in collapsed buildings during urban SAR (Edlinger et al., 2013).

Correspondingly, the deployment of these platforms has become possible as manufacturers offer a range of robust UGV platforms that can be applied in disaster response. For example, Boston Dynamics[1] provides legged unmanned ground platforms able to tackle rough terrain conditions (LS3 or the humanoid Atlas used in DARPA Robotics Challenge), while QinetiQ North America[2] provides TALON® Responder UGV platform especially designed for disaster response purposes. UGVs were deployed in disaster response in the 2001 World Trade Center tragedy and several other disasters in the following years, but due to the unreliability of the platforms the SAR teams in the United States avoided using them in the following years (Carlson and Murphy, 2005). Learning

[1]`www.bostondynamics.com`
[2]`www.qinetiq-na.com`

Manufacturer	weControl `www.wecontrol.ch`	ING Robotic Aviation `http://ingrobotic.com/`	senseFly `www.sensefly.com`
Name	CARD CH	Serenity	eBee
Wingspan	3 m	3.3 m	0.96 m
Top Speed	100 km/h	80 km/h	90 km/h
Endurance	2 hours	8+ hours	50 minutes
Max. Payload	1.1 kg	5 kg	none
Operators	two	one	autonomous

Table 1.1: Fixed-wing UAV platforms

from this, SAR teams prefer to use simplistic robots with fewer points of failure such as the Active Scope Camera (Hatazaki et al., 2007). Larger, more complex robots with complex sensors and actuators are usually applied in less time-critical situations such as the assessment of an injured nuclear reactor in Fukushima (Nagatani et al., 2013) three months after the disaster. To date, UGV platforms incorporate a minimal level of autonomy, as they are typically teleoperated and moving slowly. This is the reason why this work focuses on utilising UAVs rather than UGVs.

A wide range of commercial UAV platforms are available for supporting disaster relief. Several examples of commercial UAV platforms are compared in Tables 1.1 and 1.2. In particular, Table 1.1 compares three commercial fixed-wing UAV platforms for disaster response. As we can see, different platforms require different number of operators. Traditional manned or unmanned flight platforms require two operators: one piloting the aircraft and another handling sensor/camera readings. In contrast, modern unmanned aviation systems offer solutions to reduce the number of operators for each UAV. In more detail, autopilot systems can replace the pilot operator (especially for light-weight rotary-wing platforms where no safety pilot is required), and automatic tracking or mapping functionality can eliminate the need for a second operator. For specific tasks, such as mapping of a given area, the senseFly eBee offers an autonomous solution that requires nothing but throwing it in the air.

Besides fixed-wing UAVs, rotary-wing UAV platforms represent another product range. Their different build allows them to be used for different purposes other than fixed-wing UAVs. Table 1.2 presents small rotary-wing platforms that are easy to set up and safe to operate in urban environments, making them ideal for disaster response purposes. All these rotary-wing platforms are lightweight and can be carried, deployed and operated by a single first responder. In general, fixed-wing UAVs have longer endurance, higher speed and higher cruise height than small rotary-wing UAVs, while rotary-wing UAVs have the advantage of hovering and approaching an object from closer. In general, these UAV platforms can be used to take aerial imagery conveying crucial information for

Manufacturer	senseFly `www.sensefly.com`	Sky-Watch `http://sky-watch.com/`	DJI `www.dji.com`
Name	Albris	HUGINN X1	Phantom 3 Prof.
Weight	1.8 kg	1.4 kg	1.3 kg
Top Speed	43 km/h	22 km/h	58 km/h
Endurance	22 minutes	25 minutes	23 minutes
Foldable	no	yes	no
Operators	one	one	one or two

Table 1.2: Rotary-wing UAV platforms

first response about the disaster site. In particular, fixed-wing platforms are capable of quick, high-altitude mapping, while rotary-wing platforms can approach specific targets transmitting high-detail video footage.

As the spectrum of commercially available UAVs widens, they become increasingly used in disaster response. Applications range from mapping cluttered environments from aerial imagery (Angermann et al., 2012; Ferworn et al., 2011), to measuring radiation levels before deploying SAR teams (Han et al., 2013). Moreover, UAVs can help search and rescue in very diverse environments such as alpine rescue (Marconi et al., 2013) or indoor exploration (Luo et al., 2011). Use cases for acquiring aerial imagery with UAVs can be found in real world scenarios during disaster response (Qi et al., 2016; Syouryuu, 2016; Jones, 2014b; Meier, 2013). Also, off-the-shelf platforms were recently used for high resolution mapping of the aftermath of the earthquake in Haiti (Lubrano, 2013).

In order to extend these applications with a heterogeneous team of UAVs, making best use of the advantages of different platforms, the integration into the current practice of first responder professionals needs to be taken into account. This is a very difficult task from a research perspective. Goodrich et al. (2008) show an example for this, when they integrate a micro UAV into a wilderness SAR process. They conclude that sometimes features thought to be unimportant make a large impact on the system efficiency. This is because only the professionals will have the domain knowledge to recognise what is really important for a successful integration. For this reason, we intend to give control to first responders who supervise our proposed system, while also avoiding the workload that a manual system would generate. This can be achieved using flexible autonomy. This concept is applied in the disaster response domain in Ramchurn et al. (2015). They show that people tend to make manual changes as they lose the trust of the system when a UAV malfunctions. Accordingly, we also involve first responders in the process, as they will have better contextual information than a computer algorithm, and also to increase their trust in the system. Another important point is that we provide a system for information collection before the classical SAR process is initiated (or in the beginning of it) rather than actively involving UAVs in the SAR work. This way the SAR process in not held up when parts of the system malfunctions, as it happened during the response after the World Trade Center tragedy (Carlson and Murphy, 2005).

To that end, we envision a scenario using autonomous fixed-wing and rotary-wing UAVs, while involving first responders in the process of collecting important information for SAR teams. This scenario is the result of our discussions with Rescue Global, an international charity who makes use of the latest technologies during their disaster response activities (Rescue Global, 2013). The scenario tries to make best use of the capabilities of the different platforms and first responder professionals. The scenario does not include Gold command, as their operation (resource planing and management) is outside of its scope. The steps of the scenario are the following:

1. Bronze commanders set up and launch fixed-wing and rotary-wing UAV platforms, and start approaching the disaster site;
2. Making use of their high speed, fixed-wing UAVs start rapid mapping of the area by taking aerial images;
3. Images taken by the fixed-wing UAVs above the disaster site are transmitted to the local headquarters;
4. Silver commanders seek and locate critical locations on these images where rescue operations might be necessary;
5. Rotary-wing UAVs are automatically assigned to approach, and provide live video of these locations for silver commanders to investigate;
6. Bronze commanders are assigned to tasks according to the video footage by the silver command;
7. Bronze commanders arrive to task locations and deliver the required tasks.

As we can see in the above scenario, the critical locations for rotary-wing UAVs to visit come as a result of the mapping by the fixed-wing UAVs. This means there are dependencies between the activities of the two kinds of aircraft that develop a *workflow* structure. The workflow represents the required order in which the activities should happen. For example, in the above scenario, a high altitude aerial photo has to be taken of an area first before a rotary wing UAV could approach any critical location inside that area, or the tasks need to be determined from the rotary-wing UAV footage for bronze commanders before they could deliver these tasks. A very similar structure can be observed during urban search and rescue, when search identifies rescue tasks at a disaster site, but workflows determine the complex structure in many areas of crisis response (Wood et al., 2012).

This workflow structure can be broken down into relations between individual actions, as one action can deliver information about another. For example, taking an aerial photo by a fixed-wing UAV might uncover the placement of a critical location that has to be visited by a rotary-wing UAV, or the information gathered using the rotary-wing UAVs could result in a task that requires some special equipment only some bronze commanders carry, or might not result in a task at all. In order to find the best actions of the UAVs, the possible relations between future activities have to be considered. In the current context, this means that rotary-wing UAVs should not try to approach areas that are going to be visited much later by the fixed wing aircraft, or that rotary-wing UAVs could benefit from prioritising critical locations that can be reached soon by the bronze commanders.

In a more abstract definition, the problem of *collaboration* between rotary and fixed-wing UAVs in the above scenario can be phrased as collaboration between two distinct group of agents. One agent group explores an area with scattered critical locations (search agents), while the other group visits these locations (rescue agents). This means that the problem is collaboration between *heterogeneous* agents that play a different role in a specific workflow structure. The current literature either concentrates on searching an area using mobile sensors (Bernardini et al., 2013; Waharte and Trigoni, 2010; Gan et al., 2012; Moorehead et al., 2001) or assigning identified tasks to robots (Fave et al., 2012; Chandler et al., 2002; Gerkey and Mataric, 2002; Pujol-Gonzalez et al., 2013; Nanjanath and Gini, 2010). Some work involves collaboration of more kinds of agents for exploration (Grocholsky et al., 2006; Boumghar and Lacroix, 2011; Zivan and Sycara, 2010). These approaches involve either independent planning of the agent groups, or very simple collaborative planning of those. However, the main hypothesis of this work is *if we allow the agent groups' plans to influence each other, they will result in a more efficient first response operation.* Expanding on this, search agents knowing what areas rescue agents are planning to visit can alter the search to discover rescue tasks at those locations earlier. In a similar fashion, rescue agents can take the current plan of the search agents to know when would rescue tasks be discovered at different locations. As a result, rescue operations are delivered earlier, meaning the probability of successful rescues will increase because of the time-critical nature of SAR.

In the following, the problem formulation of collaborative search and rescue in a disaster setting is detailed.

1.3 Problem Formulation

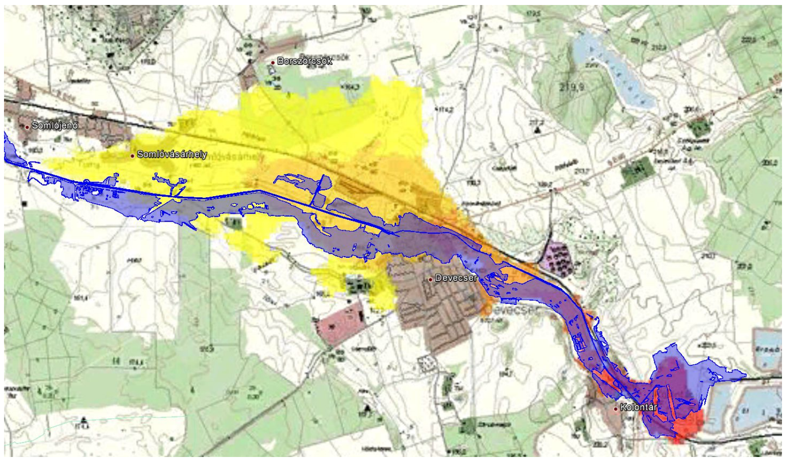
In this work we study the collaboration of a heterogeneous multiagent system that splits up into a group of search agents and a group of rescue agents. Search agents find task locations for the rescue agents during the exploration process. The problem is to determine the agents' actions based on the current knowledge about the disaster site, and maximise the efficiency of the search and rescue by performing more tasks as early as possible.

The knowledge about the disaster site, as explained in Section 1.1, can come from several sources meaning that these sources have to be aggregated. In many cases, there is not enough information available to exactly locate or identify tasks, as a report about a task might be inaccurate or there is only information about the impact of a disaster (e.g. flooded areas, population density, or path of a hurricane). As our target is information collection, there is very little known about each task before the information is collected other than its location. Accordingly, the most important property of a task in this context is its location. In order to represent the spatial information of possible rescue tasks, a probability distribution can be constructed over the possible locations using the uncertain information about a task's location. For 2-dimensional locations, this means that we represent our knowledge about possible tasks through an *intensity map*. Such intensity maps represent the distribution of tasks over the area according to the aggregated information about the disaster site. Of course, this information dynamically changes according to Section 1.1 constantly adjusting the intensity map as well.

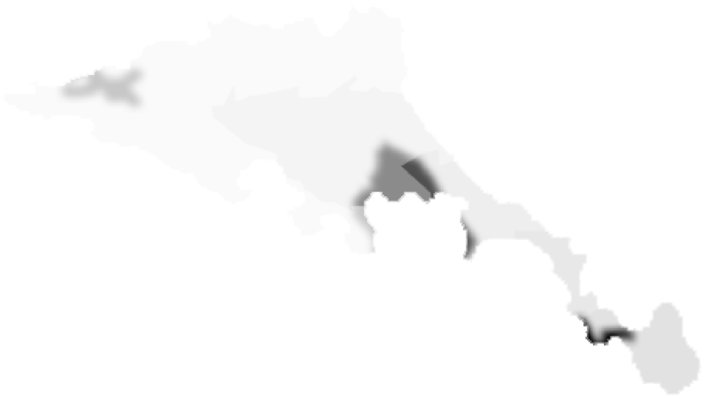
For example, fusion of different information sources into an intensity map at the industrial spill near Ajka (mentioned in Section 1.1) can be seen in Figure 1.1. Specifically, Figure 1.1a shows the result of a simulation to estimate the possible impact of a spill highlighted in red to yellow (Police of Hungary, 2010); while the actual flood caused by the spill in 2010 is marked with blue (Ambrus, 2011). Furthermore, Figure 1.1b combines the outcome of the simulation with the population density resulting in an impact intensity map.

To that end, the multiagent system can use the probabilistic information about the locations of rescues to maximise the efficiency of the SAR operation. This can be achieved by focusing the search on the areas that are more likely to contain rescue locations, or rescue agents can proactively approach probable rescue locations. Focusing the search will result in rescue locations being explored earlier, and rescues being performed earlier. Also, if the rescue agents approach areas where rescue locations will be explored, they will be able to reach these locations sooner.

In particular, this multiagent system is aimed to be deployed in disaster response, where the available knowledge about the disaster site might change from one minute to another (Section 1.1). Also, first responders need to have complete control over a system where UAVs are used to identify rescues to be performed by bronze commanders. As a consequence, such a system has to be responsive to changes in the information, as well as for input from silver commanders. This means that the underlying multiagent optimisation is required to run in a *real-time* manner. This limits computation time as the information has to refresh at least every few seconds. Moreover, the circumstances



(a) Simulated heat map of the sludge spill at Ajka alumina plant (red-yellow), actual flooded area (blue). (Police of Hungary, 2010; Ambrus, 2011)



(b) Simulated effect fused with population density (darker areas represent higher impact).

Figure 1.1: Information fusion for disaster management.

after disasters usually require first responder teams to deploy their own infrastructure (Jones, 2014a) so the available computational power is also limited to portable device capabilities.

Disaster settings are largely different from one disaster to another. This means a proposed system should not be restricted to operate within strict bounds, as future disaster setting may fall outside such bounds. Correspondingly, *scalability* is an important matter for the applied techniques so there is no hard limit for the number of operating UAVs or other parameters of the scenario.

Also, the reliability of robot platforms in real disaster settings are often questionable as detailed in Carlson and Murphy (2005). Therefore, a *robust* system structure is necessary that can deal with interruptions in terms of the communication or the operation of the robots.

To summarise, the problem of collaborative SAR by a team of autonomous agents in a disaster setting could be outlined by the following criteria.

CH: **Heterogeneity** Search and rescue agents have different roles in the workflow of their activities, thus have to be treated differently.

CP: **Probabilistic information** The current state of the available information can be described as distributions (or sets) of tasks and their relations. The relations describe the possible effects of the execution of each task to other tasks.

CT: **Time-criticality** As tasks contribute into carrying out successful rescues, and the probability of a successful rescue decreases over time, tasks have to be executed as soon as possible.

CD: **Decision making under uncertainty** The agents choose their actions to maximise the efficiency of the first response, according to the available information represented by a probabilistic model (CP).

Additionally, the following requirements should be fulfilled to ensure an effective system in disaster response:

RF: **Flexibility** The system has to be flexible to new information, as a significant amount of information is processed during first response. This means, that new information fed into the system should have a near immediate effect on the agents' behaviour taking the best available information into account.

RR: **Real-time** As the system aims to be deployed during a first response scenario, it needs to achieve collaboration in the multiagent system in real-time on a portable infrastructure. In order to achieve this, the optimisation has to run in the background and provide a response for possible alterations coming from silver command within a couple of seconds.

RS: **Scalability** The solution should be scalable, and not face intractable overheads for communication and computation with increased mission complexity. We consider settings intractable when parameters are several orders of magnitudes larger than the desired limit. For example days of computation time compared to real-time (RR) operation.

RG: **Graceful degradation** In a real-world application, especially in disaster response, unexpected events will occur. A proposed system has to be able to cope with unexpected failures in communication or in the system components while making best use of the remaining resources.

1.4 Research Contributions

In the context of the above criteria and requirements the contributions of this work are as follows.

1. We propose a novel method for homogeneous multi-robot task allocation given an arbitrary task distribution (CP) and continuous action space (UMRTA problem, defined in Section 3.1.2). This allows the robots to make decisions under uncertainty in a multi-robot task allocation setting (CD). Existing approaches either cannot fully capture the uncertainty or become intractable at near-continuous action spaces. Our solution offers a computationally efficient solution in both discrete and continuous action spaces compared to existing approaches (as detailed in Section 3.2.1.2). This part of the work has been published in Beck et al. (2016).

2. We offer the first online planning approach to create a *joint* plan under uncertainty (CD) in a complex collaborative SAR setting (CP, CT) for heterogeneous robots (CH) (defined in Section 3.1.3). As a result, rescue plans are made over an uncertain set of rescue locations and iteratively optimised along with search plans (as detailed in Section 3.2.3). This part of the work has also been published in Beck et al. (2016).

3. We present a novel Monte Carlo tree search technique (unpublished) that explores the decision space and the problem uncertainty at the same time. Existing approaches explore the uncertainty in a predefined way (inspecting a given number of samples), while our approach includes the uncertainty as a level of the tree and expands it dynamically. In order to achieve this, we determine the value of information for exploring the uncertainty (Section 4.2.2.2). The resulting Monte Carlo tree search planner provides an anytime approach (RR, RS) for solving the joint SAR planning problem.

4. We prove the efficiency and robustness (RG) of the joint online planning approach (Contribution 2) in a realistic setting. In detail, we present a robust system for finding victims using their mobile phones to locate them (Chapter 5). Our evaluation shows the advantages of full-horizon joint planning compared to the state-of-the-art limited horizon decomposition planning approach.

1.5 Outline of the Thesis

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

- In Chapter 2, we outline the related work and literature with a focus on the research areas mentioned in Section 1.2, Section 1.3, and against the criteria and requirements listed in Section 1.3.
- Chapter 3 presents and evaluates our online planner (Contribution 2) for the collaborative SAR problem. First, the underlying problems are defined. After that, the applied planning approaches are detailed (including Contribution 1). Finally, a simulation environment is explained where different collaboration approaches are compared in scenarios based on disaster assessment data.
- Chapter 4 introduces our novel Monte Carlo tree search planner (Contribution 3) for the same collaborative SAR problem. Specifically, the adjustments in the conventional tree structure and the tree search itself are described. The planner is evaluated in the same settings as in the previous section.
- Chapter 5 describes a realistic application of collaborative SAR planning (Contribution 4). In this scenario, a heterogeneous team of UAVs are used for automated victim search in a large-scale urban SAR setting. We present a robust, almost ready-to-deploy system and evaluate its performance in a similar setting to the previous chapters. We identify the key benefits of long-term joint planning compared to other approaches in a realistic uncertain setting.
- Chapter 6 draws conclusions about the work explained in Chapters 3, 4 and 5 followed by a discussion of topics for future work.

Chapter 2

Literature Review

As highlighted in the previous chapter, robots play a key role in disaster response. Robots can aid first responders in various ways by accelerating search as remote sensors or by actively taking part in the rescue process. In the following, we first discuss applications of robotics in disaster response in Section 2.1. After that, as this work concentrates on combined search and rescue, the literature about searching an unknown area with multiple robots is studied in Section 2.2, then existing approaches are reviewed about multi-robot rescue (performing a set of tasks with robots) in Section 2.3. Subsequently, we outline the state of the art in complex collaboration of robots, such as the combination of search and rescue, in Section 2.4. Finally we summarise the related literature in Section 2.5.

2.1 Robots in Disaster Response

Various applications of unmanned vehicles in disaster relief can be found in the literature. In particular, Ferworn et al. (2011) and Angermann et al. (2012) use micro aerial vehicles for 3D modeling of areas that are either dangerous or difficult to access. Angermann et al. assess a power plant that suffered structural damages, while Ferworn et al. assess a cluttered disaster site that simulates a collapsed building. In both cases, these studies use the imagery and sensor data from the vehicles to reconstruct a 3D model of the environment as an offline process.

Similar 3D mapping is carried out by Michael et al. (2012) using a team of a UGV and a UAV. The work presents a successful application of a human operated UGV and a semi-autonomous UAV for indoor 3D mapping of an earthquake damaged building. The UAV is sent to autonomously map areas that are inaccessible for the UGV. 3D modeling is a promising remote sensing ability that can be performed by an agent in disaster response. The resulting detailed 3D models can be used for damage assessment or to help accessing a victim. Even though the example scenario in Section 1.2 only includes taking aerial imagery by the UAVs, these tasks could easily be replaced by 3D modeling in a similar scenario. However, since this is not the main focus of this work, we focus on the simpler imagery-based mapping task.

Besides surface mapping, UAVs can be used to create a map of a single feature as well. In particular, Han et al. (2013) use radiation sensors and present 3D mapping of radiation levels using unmanned aerial vehicles. They use multiple UAVs flying in formation as a mobile sensor network. The paper presents a decentralised control structure for formation flight and trials of 3D contour mapping of radiation with two UAV platforms. This represents another remote sensing ability of a small group of UAVs collaborating that could gather essential information for disaster relief. The contour mapping of a radiation cloud can be useful when first responders operate in the proximity of a radiation cloud (Fischer et al., 2014). A group of radiation mapping UAVs can be one of the different agent groups operating collaboratively at a disaster site, however for the reasons outlined above we assume mapping primarily using aerial imagery in Chapters 3 and 4, and using GSM localisation in Chapter 5.

Robots can also use their sensors to locate individual features rather than creating a map. As mentioned in Section 1.1, detecting mobile phones can be a very useful capability for robots. A system that can be deployed on a larger UAV is introduced in Munoz-Castaner et al. (2015), while a similar more compact system is introduced in Andryeyev et al. (2016). The two systems differ in their size and power consumption, while both are capable of locating GSM mobile phones and establishing a temporary communication channel when there is no operational mobile network. However, the smaller and lower consumption system presented in Andryeyev et al. (2016) exhibits a significant battery life decrease for a small rotary-wing UAV (it can be operated without a safety pilot, Table 1.2) due to the extra payload and power consumption. For this reason, we assume similar equipment carried by larger fixed-wing UAVs in the application presented in Chapter 5.

When transitioning such technologies into disaster response, it is very important to provide use-cases for extending the capabilities of incident responder professionals while making best use of these new tools. In this vein, an important study has been conducted in Goodrich et al. (2008) to revise the workflow during a wilderness search and rescue operation. The benefits of several important features has been studied when using imagery from a small UAV for wilderness SAR. These features include successful ways of information representation and an integration method for UAVs in the wilderness SAR process. Such integration of this work in disaster response is discussed as future work (Section 6.2.2).

Another approach for combining new technologies with the work of disaster response professionals is creating a cooperative system of robots and disaster response professionals for a specific application. This approach is represented by two projects, SUAAVE and SHERPA (Cameron et al., 2010; Marconi et al., 2013). The SUAAVE project (Cameron et al., 2010) presents several important challenges during deployment of a multi-UAV system for wilderness SAR (e.g. reliability, computational efficiency, and physical limitations). The SHERPA project (Marconi et al., 2013) focuses on mountain rescuers teaming up with different kinds of robots. Two targeted scenarios are described, namely a wilderness SAR in summer conditions and an avalanche rescue mission in winter conditions. These scenarios present heterogeneous agent collaboration (CH) challenges besides human-computer interaction problems. Marconi et al. also introduce benchmark and validation methods for the above missions. However, the given benchmarks and measures are often vague. For example, in SHERPA (2013), a distributed planning module is proposed to deal with the collaboration of different platforms in line with RG, but the given measure of success is not related to any kind of performance measure (such as CT), only verifies whether the systems can work together and if any output can be carried out. Additionally, tests using real platforms have not yet been carried out. Furthermore as described in SHERPA (2014), the Cognition Engine interprets the mission specific background knowledge in line with our aims (CP). The two projects outlined in these papers provide good examples of autonomous vehicle applications. However, neither of these projects present or evaluate existing systems that can solve the outlined challenges.

In order to manage different kinds of robots during disaster response, a control architecture is necessary that operates the vehicles and manages communication between them. In this context, Chaimowicz and Kumar (2004) present a general collaboration framework between fixed-wing UAVs, blimps, and ground vehicles. In particular, a hierarchical structure is presented between these different platforms that allows different control over different levels. The ground vehicles follow a group behaviour that allows them to stay evenly distributed within a shape (swarm behaviour), while localized by one of the blimps – similarly to Sukhatme et al. (2001). Blimps are navigated from fixed wing aircrafts that provide an overall plan for the whole system. The hierarchical separation of the agents allows the system to be scalable (RS) even when thousands of agents are present. However, it also makes maintaining system flexibility (RF) difficult, as sometimes regrouping of the vehicles becomes necessary as proposed in Chandler et al. (2002) (see Section 2.3.3). The presented framework can deal with the localisation of up to thousands of robots, however, it does not consider the collaboration of these vehicles in a complex scenario. In this work, we do not consider the localisation of the robots, as we assume each robot can control its position.

As seen above, there are various applications using robotics to aid disaster relief, and there are many other research questions raised in the topic. Specifically, Liu and Nejat

(2013) provide a summary about the current state of the art in SAR robotics research. It concludes that there has been a great interest to provide some level of autonomy for rescue robots in recent years. Firstly, important challenges of human-robot cooperation are mentioned, such as flexible autonomy in semi-autonomous control of robots. In particular, flexible autonomy means that the level of autonomy of a robot can be adjusted during the mission. This allows the system to adapt better to different command input or to the ad hoc arrival of critical information as seen in Section 1.1, RF, and Ramchurn et al. (2015). Furthermore, Liu and Nejat point out the relevant challenges in the topics of multi-robot SAR teams, such as collaboration of heterogeneous or homogeneous platforms: *multi-robot exploration* (search), and *multi-robot task allocation* (rescue). In the following sections, ongoing work is discussed that aims to tackle these problems.

2.2 Multi-Robot Search

In this section we discuss the literature related to the problem of multi-robot search only, which means that the involved robots have a single role. This does not satisfy our criterion CH, according to which the robots take different roles in a workflow process.

During multi-robot search, multiple mobile sensors traverse an area and explore it by making observations at different locations. Such search can be guided by probabilistic information about the outcome of the search (CP). The most common way to represent this probabilistic information is by using an *intensity map* (Section 1.3) over the possible locations (Goodrich et al., 2007). This is because the intensity map is an intuitive way to highlight the likely task locations or the most valuable observation locations.

Given such an intensity map, Bush et al. (2008) address the problem of finding optimal locations for independent observations. Specifically, they use a least-squares iteration method to converge to a near optimal observation set. They present entropy and root mean-squared error as a measure of information stored in the probability map. However, in our setting, the main resource limit is not the number of observations, but the time spent making the observations (CT). Because this work does not deal with minimising the travel cost between observation locations, we cannot use it for observation planning.

Another possible representation of uncertain information is a set of locations with their uncertainty regions. In this vein, Martínez-de Dios et al. (2007) show an application of fire monitoring with multiple UAVs that uses this approach. Here, observations of different agents are merged together with previous beliefs resulting in a more accurate estimate of the current state of the fire. However, this representation assumes that the exact number of tasks in known, that does not stand for urban SAR as the number of tasks is uncertain. For this reason, we use the intensity map representation of the location uncertainty.

Uncertainty can also be represented as a collection of subjective likelihood of sets of events. Specifically, Masato et al. (2009) applies the Dempster-Shafer theory (DST) to cope with uncertain information during planning. This approach is able to combine pieces of (possibly conflicting) information with different credibility values without having to determine exact probabilities. However, the current practice in SAR is to maintain a probability distribution of the victim locations (Goodrich et al., 2008), therefore we construct a probabilistic model of the uncertainty rather than using DST.

In the following we will look into specific methods for multi-robot collaborative search. The first two method categories – gradient descent and greedy methods – make a decision by maximizing a specific measure, while the other two categories – tree search and probabilistic approaches – try to find the outcomes of a sequence of decisions.

2.2.1 Gradient Descent Methods

Gradient descent methods guide the agents in a way that the direction with the highest gradient is chosen at every state. Such approaches are often used in multiagent environments (Ogren et al., 2004; Mathews et al., 2009; Olfati-Saber, 2006; Baxter et al., 2009), because of their simple and deterministic nature and the small amount of information that has to be shared among agents.

For example, information sharing is a critical bottleneck for autonomous underwater vehicle (AUV) collaboration, as ways of communication are very limited under the water surface. This makes the gradient descent method attractive in such scenarios. In particular, Belbachir et al. (2010) show a method where AUVs explore thermal sources. Their vehicles only have opportunistic communication at predefined meeting points when they share their belief map of the underwater thermals. Under these constraints, they present a thermal localization technique where the vehicles move towards the heat source direction. Fortunately, during UAV missions inter-vehicle communication is not as restricted as in underwater environments, so information sharing can be made more often allowing a stronger collaboration and correspondingly more responsive behaviour (RF).

Another advantage of the simplicity of the gradient descent method is that it is easy to combine with other strategies. In Gan et al. (2012), for example, the multi-UAV probabilistic search problem is solved with the additional feature of collision avoidance between agents. A decentralised gradient descent search is extended with an objective of keeping a safe distance from other UAVs using a multi-step look-ahead decision-making process. Give its effectiveness and simplicity, a combination of gradient descent and deterministic task allocation is used in our benchmark approaches in Chapter 3, 4, and 5.

The flexibility of the gradient descent approach allows multiple kinds of robots to collaborate during search. In Grocholsky et al. (2006) for example, a collaborative search and target localization is discussed and evaluated using a group of UAVs and UGVs. The uncertainty of the position readings are based on a sensor model for the UAV's and UGV's camera-based sensing. First UAVs identify target positions with a level of uncertainty, then UGVs collect more sensory data about the target location to reduce the uncertainty of the target positions. The UGVs are controlled to follow the gradient in the level of mutual information, while UAVs are either following a fixed discovery pattern or controlled in a similar fashion. The presented uncertainty decreasing control method is beneficial for navigating unmanned vehicles with autonomous sensors to explore an area.

Gradient descent is a simple and efficient method for guiding agents in an information field in order to reach high value areas. For this reason, we apply gradient descent in combination with task allocation during decomposition planning (Section 3.2.1.1). In particular, rescue agents follow the gradient of task intensity when no known tasks are assigned, and the search agents choose between the closest observations on a grid based on the highest task intensity.

In the following section, greedy approaches are presented, where a step-by-step maximisation of a specific measure guides the search.

2.2.2 Greedy Methods

Similarly to Gan et al. (2012), combining different goals or information sources in probabilistic search is an important challenge. Multiple information sources (Section 1.1) and different goals and priorities (e.g. rescue victims, save damaged structures, and put out fires) might be present in a complex disaster response scenario. To this end, Moorehead et al. (2001) present a greedy method to explore an area according to multiple sources of information in a generic framework. Specifically, the method computes the next sensor reading according to information gain and a traversability map.

Using a similar approach, Chung and Burdick (2008) evaluate two biologically inspired greedy search methods to decide about sensor reading locations. One is inspired by the saccadic movement of the human eye and another by the flight trajectory of the Drosophila fly. Similarly to the method of Moorehead et al., the saccadic strategy makes a measurement at the location with the highest target probability. In contrast, the Drosophila-inspired method moves towards the current highest probability location in a straight line, constantly making observations. Now, while such greedy methods can lead to an efficient exploration strategy in one specific application, they are often difficult to adapt to a slightly different problem. Small variation in the setting may cause such algorithms to get stuck, or provide a bad quality solution. There are also difficulties when applying greedy methods for multiagent collaboration scenarios because agents may get stuck in similar states resulting in really similar behaviour. In this case, the information gathered by the agents in the similar state would not bring significantly more information than a single agent in that state, but would use multiple agent resources. On the other hand, when greedy methods are applied in a team level as opposed to an agent level, the most beneficial actions are chosen for the group resulting no conflict among the agents' actions. For this reason, in this work greedy methods are applied at a group level, producing consistently different behaviour by the different members.

To tackle this challenge, Burgard et al. (2000) present an approach to conduct organized search with multiple robots. Their approach takes into account the assigned observations of the other team members to find the most valuable next observation for each robot. This means that the value of taking an observation near a planned observation of a team member is reduced, influencing the single step ahead planner of each robot.

Greedy approaches represent a simple and efficient technique when there is a lack of an information field. This is the case for the task allocation problems, therefore we apply greedy approaches for sequential task allocation in Chapter 3. In particular, the planners for task allocation (Section 3.2.1) use a greedy sequential scheduling approach. Similarly, the rescue-aware search planner (Section 3.2.2) is based on a greedy sequential scheduling approach.

In contrast to the approaches in the last two sections, the following methods explore the outcome of robot actions multiple steps ahead.

2.2.3 Tree-Search Planner-based Methods

The previous approaches (gradient descent and greedy) use a heuristic to choose an action to converge to a good quality solution. In the case of more difficult problems, the best solution is often reached by not choosing the most beneficial current action, but an action that contributes in taking further actions that will increase the overall solution quality. This means that planning multiple steps ahead can be beneficial in many cases. Taking different actions after each other can be represented as a tree, and finding the best action plan can be represented as a search on this tree. The most commonly used tree search method for this problem is A* (A-star) (Hart et al., 1968), where each node provides a heuristic that gives a lower bound for the utility of the solutions that can be found in the corresponding subtree. Using such a heuristic, the optimality of the solution (a leaf node in the tree) can be assured.

There are many different approaches using the A* search for planning multiple steps ahead. For example in Luotsinen et al. (2004), each agent chooses an available frontier location of the explored area and expands the exploration by observing at that point. The A* search path planner is then used to evaluate the distance from the available frontier points and the algorithm chooses the closest one. The presented solution deals with uncertainty in a discrete way. In particular, every grid point of the map can be either unknown, occupied or accessible. However, this simplification significantly restricts the representation of uncertain information (CP), as the probability distributions extracted from background information cannot be used in this approach.

In contrast, Bernardini et al. (2013) present an approach that deals with uncertainty in a less discretised manner. This aims to search and track a target using a UAV platform. Once it loses track of a target, it conducts a probabilistic search to find it. The search is assembled from multiple predefined search patterns that are scheduled to maximise the probability of finding the target, taking the motion model of the target into account. The scheduling is carried out by an anytime weighted A* planner that is well suited to real-time applications (RR). However, the strategy only involves a single UAV searching for a single target, and similar planning for multiple agents and multiple targets is very difficult due to the rapidly expanding decision space (RS). A good example for this is how off the shelf temporal planner algorithms do not scale well with the number of agents for a simple search problem as reported in Coles et al. (2010). Long-term online planning in a similar setting can only be achieved after reducing the action space by discretising the problem as in Burns et al. (2012). Burns et al. also apply determinisation to cope with an uncertain planing problem.

A simple but very efficient method for determinisation is *hindsight optimisation* (HOP) (Yoon et al., 2008). This technique takes several samples of the uncertain problem, and constructs deterministic problems from these samples. By solving these deterministic problems, an upper bound of the solution quality for the uncertain problem can be produced. This is an upper bound because the solutions are produced "as if the outcome of the uncertainty would be known in hindsight". Using these "hindsight" solutions for the deterministic problem help finding valuable actions under uncertainty. Because planning under uncertainty is a very important factor in SAR (CP, CD), we use the computationally efficient method of HOP to produce online plans under uncertainty in this work (see Sections 3.2.1.2 and 4.1.2 for more details).

When using exhaustive tree search, there is no need for providing heuristics, however, the size of the tree needs to be limited. In this vein, Crispin and Sobester (2015) provide an exhaustive tree-search based approach for collecting information about atmospheric variables for multiple fixed-wing UAVs in a windy setting. As concluded in this paper, the tree branching and the search depth is a critical factor for the computational cost of the planning and also has a significant effect on the performance. Both parameters need to be kept low for the presented online planning scenario (RR) even when planning with a single agent. This means that even in the case of 3 children each node, the planner can only consider 3-4 steps ahead.

Unfortunately, the lower bound of the utility (as required by A*) can be difficult to provide because of the complex time and inter-agent dependency in a collaborative SAR setting. This means that applying an efficient A* tree search on the large decision tree of the multi-agent search problem is not computationally feasible. Also, a tractable exhaustive tree search would only produce a very limited plan (highly discretised and small number of planning steps).

Once the decision tree gets larger, exploring it in a probabilistic way can be efficient. For this reason, approaches for evaluating possible decisions as a result of random sampling are discussed in the following subsection.

2.2.4 Probabilistic Methods

Solving collaborative SAR optimally is intractable[1], because the joint decisions of the multiple agents generate an enormous decision space that provides different utility over the different outcomes of the uncertain process of search (CD). For this reason, it is beneficial to look into probabilistic approaches that randomly explore the outcomes of different possible decisions, as the computation time will not be nearly enough for finding the optimal solution (RR). In particular, these techniques either use a decision tree representation of the problem or a Markov decision process representation.

The most popular approach for probabilistic searching of a decision tree is *Monte Carlo tree search* (MTCS) (Browne et al., 2012). This does not explore the complete decision tree, but constantly expands it with random leaves at the most promising locations. Each iteration contains the following steps:

1. *Selection*: The most urgent expandable node is selected by descending though the decision tree using a Selection Policy.
2. *Expansion*: One (or more) random children nodes are added to the selected node.
3. *Simulation*: The value of the added node is determined by simulating an outcome using the *Default Policy.*
4. *Backpropagation*: The values of the parents of the selected node are updated according to the new nodes.

These steps can be grouped into two policies:

1. *Tree Policy*: Select or create a leaf node from the nodes already contained in the search tree (*Selection, Expansion*).
2. *Default Policy*: Simulate an outcome from a given state to produce a value estimate (*Simulation*).

[1]Our initial tests show that when seeking the optimal solution for a small example problem (two search agents, 7 search tasks, 3 rescue agents, 12 certain rescue tasks) of the problem defined in Section 3.1.3 did not finish within 2 days using IBM CPLEX.

The *Backpropagation* step does not use a policy itself, it updates the node statistics of the ancestors of the newly added node. After the iteration process is interrupted (the computation budget is reached), the best action is determined according to the current state of the tree (*winning action selection*). This process results in an anytime solver that is important for real-time applications (RR) and the dynamic expansion allows optimisation over a highly branching decision tree. The problem of collaborative SAR using robots can be modeled as a wide decision tree (detailed in Section 4.1), for this reason MCTS can be used to find a good solution using a limited computation time (RR, RS). To this end, an MCTS-based method for collaborative SAR is detailed in Chapter 4.

As described in Browne et al. (2012), MCTS made a breakthrough in artificial intelligence for the complex two player game of Go. Besides game playing, MTCS has various applications. A version of MCTS, *rapidly exploring random tree* (RRT), is claimed to be very powerful for path planning over complex domains (Lavalle, 1998). This could be for example path planning of a 6 DOF robotic arm in a cluttered environment, or path planning of a mobile sensor over a complex utility function. However, it cannot cope with a complex collaboration problem under uncertainty such as SAR, because it does not capture the temporal constraints of the planning well. We evaluate the performance of an RRT-based path planner in Chapter 5 and find that long-term task-based approaches are more beneficial for search in SAR settings.

Using this technique, Gan and Sukkarieh (2011) use multiple UAVs in a probabilistic target search setting. As discussed in this work, RRT is efficient for path planning for individual robots, but when the objective defined as the joint actions of the robots different actions become beneficial. For this purpose, decentralised decision making and an explicit gradient model is utilised to optimise the planned path of each robot in the context of other robots' planned paths. They show the computational advantage of their explicit gradient method compared to the finite differencing approach. In contrast, Scerri et al. (2007) only rely on RRT path planning in a similar probabilistic multi-UAV target search problem. The plans of other robots are directly fed into the RRT planning process during their radio emitters location scenario. The agents share a cost map related to the entropy in the environment and use a modified RRT planner to find their path to maximise the expected change in the entropy. This way this approach does not only minimise the non-detection likelihood as in Gan and Sukkarieh (2011), but increases the accuracy of detected targets. For this reason, we apply their approach as a benchmark for the mobile phone search problem (almost identical problem to locating RF emitters) in Chapter 5.

The MCTS approach has been also used for online planning under uncertainty in Bjarnason et al. (2009). This approach combined HOP and MCTS to plan under uncertainty, evaluating a number of combined approaches for the game of Klondike Solitaire. However, there are several parameters left to be tuned in order to optimise the performance.

We also apply a combined MCTS and HOP approach in Chapter 4. We provide an extension by devising probabilistic heuristics for making such decisions that are typically adjusted by parameters in Bjarnason et al. (2009).

A more suitable probabilistic solution for a discrete decision space is modeling it as a *partially observable Markov decision process* (POMDP) (Monahan, 1982). A POMDP is described by a set of states, an initial belief state distribution, a set of actions, transitional probabilities between states when taking specific actions, a set of observations, observation probabilities, and a reward function for executing a specific action in a specific state. The goal is to find a set of actions that maximises the expected sum of rewards over an infinite horizon using a discount factor for later rewards.

There is a decentralised variant called Dec-POMDP (Amato, 2015). Similarly to the definition of POMDP, it is represented by the $\langle I, S, b_0, \{A_i\}, P, \{\Omega_i\}, O, R, \gamma \rangle$ n-tuple, where I represents the set of agents, and the actions (A_i) and observations (Ω_i) are defined as separate sets for each agent. Consequently, the transition probability (P), the observation probability (O), and the reward function (R) is defined over joint actions ($\vec{a}$) and joint observations ($\vec{o}$). This approach provides a more robust solution by planning with each agent, not relying on an uninterrupted communication with a central planner unit (RG). However, this could end up with a large communication overhead, therefore the cost of communication should be incorporated in the planning process (Gasparini et al., 2016).

Using the POMDP approach, Waharte and Trigoni (2010) present a victim search scenario with obstacles that can obscure the target detection. In particular, they compare random, greedy, greedy look-ahead, potential field based, and POMDP approaches in a simulated scenario. The problem is discretized, as the UAVs can only move and make observations on a grid. They show that the POMDP approach performs well in the given scenario, outperforming potential field based approaches, and performs very similarly to a greedy look-ahead strategy. Speaking more generally, POMDP provides effective approaches for general probabilistic decision making, but it has scalability issues (RS) (Schesvold et al., 2003). In particular, as the decision space increases with the number of tasks, agents, or the resolution of the grid, the POMDP problem quickly becomes intractable. And once the problem is simplified to a level where solving the POMDP is feasible, greedy methods tailored to the simple problem will perform just as well. For this reason, we do not consider the computationally expensive POMDP-based solution of collaborative SAR.

In the following section we review the literature on the other part of the collaboration problem, the rescue operation.

2.3 Multi-Robot Rescue

In contrast to the previous section dealing with search, this section presents methods for the rescue part of the collaborative SAR problem only. In this setting there is no uncertain information taken into account in the decision making, against criterion CD.

Multi-robot rescue consists of carrying out rescue tasks using a group of mobile robots, so it can be formulated as a *multi-robot task allocation* (MRTA) problem (Gerkey and Mataric, 2003). This formulation captures the most important feature of the problem that is the different cost associated with traversing among tasks with the agents depending on a distance metric. There are several different approaches to enable the collaboration between the robots that are broken into four categories based on the way the collaboration method is distributed in the following sections. In Section 2.3.1, human operators conduct the collaboration, so there is no automation present for this purpose. After that in Section 2.3.2, a centralised planner carries out the collaboration. Then in Section 2.3.3, the collaboration is achieved via distributed negotiation (auctions). Finally in Section 2.3.4, the collaboration is formalised as a distributed optimisation problem.

2.3.1 Human-Operated Collaboration

Many examples demonstrate that human operators handle complex tasks very well while teleoperating unmanned vehicles (Sukhatme et al., 2001; Martínez-de Dios et al., 2007; Liu and Nejat, 2013), but the forms of collaboration between multiple operators is rarely evaluated. To this end, Perkins and Murphy (2013) conducted a trial at a disaster response training ground to evaluate the performance of collaborating first responders who were teleoperating robots. Several hierarchies are tested for teleoperating a UGV and a UAV by a first responder group including classical mediated cooperation and active cooperation introducing a facilitator role that focuses on the collaboration by keeping contact with all participants. It concludes that having a facilitator between the UAV and UGV operator groups improves the overall efficiency, as the mission time decreased. It is also noted that the ability to see the video feed from other robots further improves the collaboration. Unfortunately as the number of vehicles increase, information sharing (e.g. video feed or action plan) can easily become a distraction or source of confusion for the operators (Chen et al., 2011), as well as the facilitator role becoming more complex (e.g. deciding which vehicles collaborate, multiple collaboration at the same time). For these reasons, relying only on human communication for conducting collaboration does not result in a scalable (RS) approach. This supports the need for a system with a certain level of autonomy of robot groups for a larger scale disaster response mission (Murphy, 2011) as outlined in Section 1.2. For this reason, this work and the following sections deal with making autonomous decisions with the robots rather than relying only on human guidance.

2.3.2 Central Planner-based Collaboration

Similarly to the architecture of human guided first response in the previous approach, the following solutions all include a central intelligent planner that defines the behaviour of the vehicles. In the scope of mathematical planning, the *team orienteering problem* approach (Chao et al., 1996) describes a very similar problem to MRTA. In this problem multiple agents walk cycles on the edges of a graph maximizing the utility gained by visiting the graph nodes, while the cycle size is constrained by a maximum sum of the edge weights. In the case of MRTA, the nodes represent tasks, and edges represent paths between tasks. The cycle size constraint represents the battery life of a robot. For such problems, Vansteenwegen et al. (2011) introduce the state of the art solvers for various versions of the orienteering problem. Most importantly it compares different approaches for the team orienteering problem (including tabu search, variable neighbourhood search, ant colony optimisation, and greedy randomised adaptive search). As it concludes, neighbourhood search methods happen to solve the problem the most efficiently for larger instances, because the method scales well with the problem size (RS).

Neighbourhood search techniques (Hansen et al., 2010) consist of iterating several simple steps to find a local maximum utility solution in the solution space, and then shaking the solutions (randomly moving them in the decision space) to find another local maximum. Repeating these steps, it converges to a solution representing the global maximum. This technique starts with an initial solution based on a heuristic. In line with our problem definition, we use a computationally efficient shortest adjusted processing time first (SAPT) algorithm (Rabadi et al., 2007), that is often used as a heuristic in the scheduling literature by for example Rabadi et al. (2004) and Radhakrishnan and Ventura (2000) for the same problem. Using this initial solution, we use a range of techniques to improve it in the following chapters.

Simulated annealing applies the same principle as neighbourhood search. In this vein, Leary et al. (2011) uses a simulated annealing method for multi-robot rescue. It compares three different approaches for multi-UAV mission planning with tasks and no-fly zones: mixed integer linear programming, simulated annealing, and distributed negotiation via bidding. It concludes that the simulated annealing approach manages to find the best quality solution, but it takes significantly longer than the bidding process. For this reason, we do not use a method that runs until a convergence criteria is met (such as simulated annealing), as the computation time is restricted (RR).

As mentioned above, *mixed integer linear programming* (MILP) is also an attractive approach for MRTA. It offers an optimal solution using a general solver. The problem is formulated as a set of constraints that has to be fulfilled and a measure that can be maximised. The solution space is represented as a set of decision variables. In case of task allocation, these variables represent the allocation of agents, thus they can take a discrete set of values. The MILP solver can be used in several ways for multi-UAV coordination. In particular, Furukawa et al. (2003) concentrate on multi-UAV coordination in a time-optimal manner. They use the kinematic models of fixed-wing aircraft and control them to approach multiple military targets, using MILP to find the best task allocation. Using similar techniques, Bellingham et al. (2003) coordinate UAVs while avoiding obstacles. They devise an efficient method for obstacle avoiding path planning among tasks, and feed these path lengths into the MILP solver to find an optimal allocation of the UAVs for minimal mission time. Another military application is described in Spry et al. (2005). They deal with the problem of convoy protection using fixed-wing UAVs. Specifically, they present different flight paths for providing video footage of a moving convoy. The different tasks in this case are flight patterns above certain convoy units. However, the problem with using MILP for MRTA is the lack of scalability (RS). The decision space explodes as the number of tasks or agents increase, meaning the solver takes excessive time to find the optimal solution. An example of this behaviour is shown in Leary et al. (2011). Our initial tests showed the same about the run time of the MILP planner that would not allow us to achieve the planning within the computation time limit (RR).

In more complex scenarios, as different types of tasks have to be delivered, the central planner has to solve a more difficult problem than MRTA. To this end, Rasche et al. (2010) present a combined approach, presenting different tasks for an agent. In this scenario, at first agents explore an unknown terrain, and then they have to reach certain goal locations. The exploration is completed using a potential field approach, using attractive and repulsive force fields. An A* planner (Section 2.2.3) is designed to help out agents that are trapped in a local minima during exploration or to navigate optimally towards a set of given goals. These goals of the agents are set by an external task allocation algorithm. These different roles are particularly important for a heterogeneous multiagent team, as different agents with different capabilities might perform better in different roles (CH). The solution approach shows a simple potential field approach that keeps computation time low (fulfilling RF, RR, and RS). This approach shows a way to adapt the task allocation algorithm to probabilistic information, similar to CP. However, this approach does not capture the dependencies between the actions in a search and rescue mission. Therefore it cannot be used for collaborative SAR.

On the other hand, Doherty and Rudol (2007) consider the case where two different kinds of tasks are completed with dependencies between them. Their approach includes autonomous exploration of a disaster area while searching for victims, and then delivering medical packages to these identified victims. The two different tasks are executed in two different stages. The scenario presented in Section 1.2 is broadly based on the process described by Doherty and Rudol. More specifically, the package delivery tasks are created as a result of an exploration process in their work, similarly to the rotary-wing UAV tasks in our scenario. However, their solution of separating the two processes does not involve any collaboration between the agents delivering the two kinds of activities. This means there is scope for enhancing the response time (CT) by making the execution parallel, as per our scenario in Section 1.2.

Another example for a similar workflow task structure is the problem of military target reconnaissance, attack and damage evaluation. These three tasks have to be delivered on each target in a strict order by possibly different agents. In this vein, Jing et al. (2009) present a solution based on a differential evolution algorithm that computes the best plan. The planning is separated by the three task types as hierarchical levels. Thus, plans are created for one level after another, considering the previous level plans given. This represents one of our approaches described in Section 3.2.2, the sequential planning along the workflow structure. However, their solution cannot plan under uncertainty, therefore we did not consider their differential evolution approach.

The other possibility of segmentation of the problem is by individual agents. To this end, Kvarnström (2010) introduces a planner designed for multiagent temporal planning. It uses forward chaining to plan for individual agents and then joins the plans together. While the approach presented in Jing et al. only enables a one-way collaboration along the hierarchy, Kvarnström's planner enables a more flexible collaboration between agents, by having effects propagating backwards along dependencies. However, temporal planners cannot capture the uncertainty of the problem due to its structure, therefore cannot be used for uncertain action sets (CP). For this reason, temporal planners cannot be used for collaborative SAR planning.

2.3.3 Distributed Negotiation

The previous works were based on a central planner that organised the agents' plans and decisions, thus creating a centralised system. A much more robust system can be obtained when the decision making is decentralised between the different robots (RG). To this end, the *contract net protocol* (CNP) (Smith, 1980) is the earliest general distributed negotiation technique. In this approach, the decision process is started by a manager issuing a call for proposals. Agents submit their proposals, which can be accepted or rejected by the manager. If a proposal is accepted, the agent processes the subject of the proposal, and either manages to reach the goal or fails, which often results in a repeated call for proposals.

To date, the CNP has been successfully deployed in the field of multi-robot rescue (Gerkey and Mataric, 2002; Lemaire et al., 2004). In both works, the role of the manager is flexible and distributed between the agents, resulting in a true decentralised system without a single point of failure (RG). Gerkey and Mataric (2002) allocate randomly appearing tasks to the most suitable available robot as a task appears. There is no task reassignment implemented once a task has been allocated. However, for this reason this approach may respond inefficiently if multiple tasks appear in a short time interval. Thus, the response is not flexible in the presence of incoming information (RF). On the other hand, Lemaire et al. present a method for allocating multiple tasks between the agents. It builds up the task allocation gradually, taking the individual agent's workload into consideration, which improves the overall plan efficiency. Besides that, it presents a way to handle tasks where there are inter dependencies in a distributed way. Specifically, a temporary hierarchy is constructed between the agents with task dependencies. As discussed in Section 1.2, such task dependencies are essential in a workflow task structure that are common in collaborative search and rescue. Unfortunately, while CNP provides a way to create a robust system by decentralising the collaboration (RG), it cannot handle major changes in the available information (RF) due to its static structure. Moreover, it does not capture uncertain information (CP), as it is based on static tasks.

Other solutions using inter-agent auctions are presented in Chandler et al. (2002) and Nanjanath and Gini (2010). In these approaches, agents place bids in auctions, but the structure does not follow the CNP. In particular, Chandler et al. present a system for military target classification attack and damage assessment using UAVs. They use a linear program (Section 2.3.2) for task allocation in a smaller group, and the auction process helps to allocate tasks between these groups. Dividing the agents into subgroups results in a scalable solution (RS), but also brings up a problem of the dynamic grouping of agents that is not solved in their work. If the grouping remains static, the system becomes less flexible (RF) for unexpected changes in the environment.

In contrast, Nanjanath and Gini (2010) adopt a more flexible, repeated auction architecture for MRTA. Their auction structure finds a close-to-optimal task allocation for the team by successive single item auctions. The solution overcomes the rigid single auction planning, and repeats auctions from time to time so it can react to robot failures or environment changes (thus, fulfilling RF and RG). However, their auction method considers homogeneous tasks with no dependencies defined between them, and the extension to more complex task structures (CH) is unclear.

Such task dependencies can be captured by building a tree structure of tasks where dependent tasks are represented as different leaves of the same parent node. To this end, Cao et al. (2010) describe a decentralised auction-based technique for MRTA of such a task structure. In this approach, agents can place bids on any branch of tasks. Unfortunately in the case of the collaborative search and rescue scenario, tasks at different stages of the workflow have to be carried out by different agents. This means that a single agent is not allowed to execute a complete branch of the tree according to the presented task structure, thus placing bids on branches becomes pointless for this kind of collaboration.

Liu et al. (2013) offer a similar auction-based decentralised task allocation for complex task trees. In their approach, communication constraints are introduced between certain robots. They offer a robust technique where robots can place bids according to their acquaintances' abilities. The introduced technique handles limited communication between robots very well based on field experiments, and therefore meets RG. This is very beneficial when deploying a very diverse team of robots with different sensors and actuators (typical for ground robots). However, the outlined scenario in Section 1.2 describes a system using robots with restricted diversity. Fewer types of robots makes the system maintenance easier as well as it helps recovery from robot faults (RG). In scenarios with a few types of robots, their method simplifies to a greedy, instantaneous assignment MRTA (Gerkey and Matarić, 2004). In contrast, we consider time-extended assignment MRTA in our planning problem (Section 3.1.1) that extends instantaneous assignment with the ability to produce long-term plans.

A different approach is presented in Le et al. (2009). In this case, agents are associated with different resources that use a knapsack-based algorithm, GAP-E, to allocate sensors to tasks. The resources are allocated in an iterative fashion, iteratively updating a cost and an importance matrix of their combined allocation. The system is able to create automatic allocation in a large pool of resources (RS). This approach works well when a complex and diverse combination of resources is necessary for the completion of tasks such as in the described intelligence, surveillance and reconnaissance context. Our SAR scenario however, concentrates on the long-term planning of the robots' collaboration under uncertainty (CD) rather than the joint allocation of different resource types to deterministic tasks. A similar approach would become trivial in the collaborative SAR context, as each task requires a single sensor and sensors cannot be shared between tasks.

2.3.4 Distributed Optimization

Compared to the distributed negotiation process, distributed optimisation offers a more flexible approach for complex optimisation between agents. Instead of the simple roles of bidders and auctioneers of the agents in distributed negotiation, distributed optimisation can have a more complex structure that enables more advanced optimisation by message passing techniques. To achieve this, the problem needs to be formulated as a distributed constraint optimization (DCOP) problem (Yokoo, 2012). In this case, agents distributedly choose values for variables to maximise or minimise an overall cost in the system while fulfilling a set of constraints. This results a very flexible system (RF) where any change in the available information propagates through the system resulting in an immediate change in the agents' behaviour. In particular, approaches based on the generalized distributive law (GDL), more specifically max-sum (Farinelli et al., 2008), are well suited to the problem of MRTA, where the decisions represent the allocated tasks for agents. To this end, Macarthur et al. (2011) introduce several improvements of the max-sum algorithm that decrease overall calculation time and improve solution quality compared to the classical max-sum implementations. However, the presented problem relies on the independence of task utilities which is not true for MRTA in a time-critical system (CT).

Building on the same basis, Fave et al. (2012) show an application for using max-sum for multi-UAV task allocation. Their solution associates a variable with each UAV and a constraint (function) with every task. The tasks involve several UAVs arriving to a location and providing aerial imagery. In this case, the utility associated to a task correlates with the probability of the accomplishment of the surveillance task. A task's duration is represented as an exponential random variable, and the only way a task will not be completed is due to one of the UAVs running out of battery before the task is completed. However, the task allocation method can only allocate one task to an agent. This approach can result in sub-optimal solutions when the UAVs are capable of completing several tasks in a row, as they might be allocated to a task that leads them away from a location with high density of tasks, so executing further tasks may become more difficult.

In contrast, Pujol-Gonzalez et al. (2013) formulate the optimisation problem in a different way. Each task can be allocated to one of the agents, but one agent can be allocated to several tasks. In the first approach, the cost of executing a task is considered independent. The second approach presents a workload-based solution, similarly to Lemaire et al. (2004), where a task executed by a more occupied agent gains less utility. In a realistic scenario, the more tasks an agent is allocated to, the later those tasks will be completed. This means the utility gain will be lower in a time-critical settings (CT). Through empirical evaluation, the method is shown to perform well against state-of-the-art auction-based algorithms. Yet, none of the presented approaches can handle inter dependencies between tasks, that is necessary for heterogeneous agents to work together in workflows (CH).

In summary, there are many ways to approach the multi-robot rescue problem, but we utilise relatively simple approaches in this work, as the main emphasis is on the collaboration between the search and rescue activities. In particular, our aim is to show that when the optimisation criteria is to improve the collaborative performance, approaches of the same complexity are able to produce a better joint plan than when the optimisation is separate for the search and rescue. The related work in this particular area can be found in the next section.

2.4 Collaboration Between Search and Rescue Robots

This section discusses different approaches for multi-robot collaboration in a setting where different robots take different roles according to CH. This is essential for delivering complex operations such as search and rescue, where the activities of the different robots create a workflow structure (as discussed in Section 1.2). In real disaster response scenarios, the information is likely to come from various sources (Section 1.1), and most of it will include a degree of uncertainty about the mission or task properties (CP). Given this, a number of solutions have been developed to establish collaboration between heterogeneous robots in uncertain environments.

One example for such collaboration is path finding for UGVs using aerial images of UAVs. To this end, Boumghar and Lacroix (2011) present a UGV using a path planner to find possible paths though the known and unknown regions of the map to reach its goal, while an aerial vehicle explores the area taking the possible UGV paths into account. This results in the UAV exploring areas that the UGV is most likely to pass through while avoiding obstacles. This is a good example for sequential planning against the workflow structure that we apply for the SAR collaboration problem (Section 3.2.2). On the other hand, collaboration between a single UGV and a supporting UAV is presented for avoiding obstacles in an unknown terrain by the UGV, the model does not extend to multiple possible goal positions that have to be reached with multiple vehicles. More specifically, the decision making of which robot to go towards which target location (multi-robot task allocation) is not considered. The way they cope with uncertainty of the obstacle locations is similar to the approach we applied for the rescue task location uncertainty. Their approach also averages multiple simulations of possible outcomes of a random process. This approach is similar to our HOP approach detailed in Section 3.2.1.2.

In contrast, Zivan and Sycara (2010) present the collaboration of multiple agents in a target search and tracking scenario. In particular, these agents split into two groups of mobile agents with different capabilities in terms of sensing and mobility. The collaboration is achieved by a decentralised optimisation algorithm using a maximum gain message (MGM) method. They present a target search and detection scenario, that is very similar to the scenario presented in Section 1.2. Specifically, one group of agents is searching an area relying on a task intensity map (CP), while members of another agent group visit the target locations pointed out by the first group to convey further information about the targets and tracking their position. The cooperation across these teams of agents is initially hierarchical: the first group searches independently from the other group, and the second group relies on the information discovered by the search. After that, they provide a way of making collaboration across the two teams stronger by sharing activities between agents with different roles and feeding back information about the targets' motion to the search agents. However, this collaboration means that the groups perform independent (decomposition) planning. As we will discuss it later, such approaches can be outperformed by collaborative planning with the agent groups as shown in Chapters 3, 4, and 5.

Finally, the Action project (Barbier et al., 2009) focuses on a general framework for collaboration between a team of heterogeneous autonomous vehicles (CH). The project addresses planning for these autonomous vehicles in a scenario where the uncertainty of information is taken into account (CP). They include task dependencies, that are very important for vehicles to work in a workflow task structure. In particular, two decentralised solution methods are proposed: one using decentralised Markov decision processes (DEC-MDP) and another using hierarchical task networks. The first is a probabilistic approach, while the second relies on auctions for a known task structure. The DEC-MDP approach cannot be applied under uncertainty in the actions (CP), and when the problem is formulated as a POMDP, the computational complexity will become unfeasible to compute in real-time (RR) as previously mentioned in Section 2.2.4. Also, the hierarchical task network approach cannot cope with uncertain tasks as discussed in Section 2.3.3. For these reasons, none of these approaches are considered for collaborative SAR.

2.5 Summary

This chapter has outlined the state of the art in multi-robot collaborative search and rescue against the criteria and requirements listed in Section 1.3. In collaborative SAR, heterogeneous agents operate with a workflow structure between their activities (CH) in a time-critical manner (CT). The information about the exact activities gradually builds up during search, while rescue covers scheduling these activities. Subsequently, it is crucial to operate on the basis of the best available information as the search proceeds (CD). This means uncertain information about the future outcome of search needs to be considered for a successful collaboration (CP). In this vein, we apply a group level negotiation-based planning using HOP (described in Section 2.2.3) for collaborative SAR and compare against the independent approaches typically used in similar problem settings (Sections 2.2.1 and 2.2.2) in Chapter 3. Furthermore in Chapter 4, we apply probabilistic tree search (Section 2.2.4) in order to produce an anytime solution for the collaborative SAR problem, that we compare against the previous approaches. We present a robust system (RG) for victim search using mobile phone detecting UAVs (Section 2.1) using the decentralised version of these approaches in Chapter 5.

In this literature review, two methodologies are identified for heterogeneous robot collaboration under uncertainty. Firstly, the *decomposition* planning where the planning is independent for the different robot groups (Zivan and Sycara, 2010; Doherty and Rudol, 2007). Secondly, *unidirectional* collaborative planning approaches. This can be done using sequential planning along the workflow structure, when agents use the plans of the plans of agents that are at an earlier stage of the workflow process (Jing et al., 2009; Lemaire et al., 2004). Alternatively, the sequential planning can be done against the workflow structure, in this case agents use the plans of other agents at a later point of the workflow (Boumghar and Lacroix, 2011). These approaches are compared against our *bidirectional* planning approaches in the following chapters.

Chapter 3

Collaborative Planning Between Search and Rescue Robots

As discussed in Chapter 1, there is significant interest in using robotics in disaster response. In particular, the need to use a larger, diverse group of robots that can be supervised by a smaller team of first responders arises after a number of initial applications of UAVs in disaster response (Lubrano, 2013; Goodrich et al., 2008). However, in order to achieve this, a greater degree of autonomy and intelligence is required from the robots than is common today (Murphy, 2011). An important aspect of this enhanced functionality is to ensure there is effective collaboration of heterogeneous robots that participate in a given incident.

To provide a specific setting for such collaboration we developed the scenario (outlined in Section 1.2) as a result of discussions with Rescue Global (Rescue Global, 2013). As concluded in the literature review (Section 2.5), there are two main types of planning architectures for tackling similar collaboration problems. In this chapter we propose a third architecture and evaluate it through simulation in two different realistic disaster settings. Specifically, the decomposition, unidirectional and our bidirectional planning approaches are compared.

The collaboration problem is defined in Section 3.1. After that, the compared planning approaches are detailed in Section 3.2. The comparison through an empirical evaluation is described in Section 3.3. Finally, the findings are summarised in Section 3.4.

3.1 The Collaboration Problem

As discussed in Chapter 1, the SAR collaboration problem involves agents of different roles (CH), contains uncertainty about tasks and relations between tasks (CP), that are time-critical (CT). In this section, we define the SAR collaboration problem that satisfies these criteria. More specifically, it involves two agent groups with different roles, referred to as search agents (or search robots) and rescue agents (or rescue robots). The problem is searching a region by visiting certain points of an area and making observations (search tasks) by the search agents. These observations result in finding rescue tasks, that need to be visited and completed by rescue agents. The sooner each rescue task is completed, the higher the utility gain associated with it.

The following example shows a possible scenario for such collaboration:

1. Figure 3.1
   Search (marked with green):
   There are two search agents S1 and S2 (green circles) and three search tasks A, B and C (green rectangles). The areas associated with the search tasks are marked with dashed rectangles. Initially S1 is allocated to complete A and S2 is allocated to complete C. Planned schedule of agents: S1 $\leftarrow \langle A, B\rangle$, S2 $\leftarrow \langle C\rangle$.
   Rescue (marked with gold):
   There are three rescue agents R1, R2 and R3 (yellow crosses) and no rescue task known initially. R1 is initially approaching A area, R2 is approaching B area, and R3 is approaching C area.

2. Figure 3.2
   Search:
   S1 has made an observation at A and found five rescue tasks (marked with crossed circles and numbers from 1 to 5). S2 has changed its allocation to complete B because the rescue agents are not approaching C (detailed below). Planned schedule of agents: S1 $\leftarrow \langle\rangle$, S2 $\leftarrow \langle B, C\rangle$.
   Rescue:
   Because of the large number of rescue tasks (5) found at A, both R1 and R2 are tasked to complete these tasks (task 5 and 4 specifically). For this reason R3 is approaching the nearer B area. Planned schedule of agents: R1 $\leftarrow \langle 5, 3, 1\rangle$, R2 $\leftarrow \langle 4, 2\rangle$.

3. Figure 3.3
   Search:
   No change, S1 is idle, S2 is approaching B. Planned schedule of agents: S1 $\leftarrow \langle\rangle$, S2 $\leftarrow \langle B, C\rangle$.
   Rescue:
   R1 has reached the location of task 5 and started completing it. Planned schedule of agents: R1 $\leftarrow \langle 5, 3, 1\rangle$, R2 $\leftarrow \langle 4, 2\rangle$.

4. Figure 3.4
   Search:
   S2 has made an observation at B and found 2 rescue tasks, 6 and 7. S2 is now

allocated to complete C. Planned schedule of agents: S1 $\leftarrow \langle\rangle$, S2 $\leftarrow \langle C\rangle$.
Rescue:
R3 is now allocated to complete 7. Planned schedule of agents: R1 $\leftarrow \langle 5,3,1\rangle$, R2 $\leftarrow \langle 4,2\rangle$, R3 $\leftarrow \langle 7,6\rangle$.

5. Figure 3.5
Search:
No change, planned schedule of agents: S1 $\leftarrow \langle\rangle$, S2 $\leftarrow \langle C\rangle$.
Rescue:
R1 has completed 5, it is now allocated to complete 3, R2 has reached 4, and R3 has reached 7. Planned schedule of agents: R1 $\leftarrow \langle 3,1\rangle$, R2 $\leftarrow \langle 4,2\rangle$, R3 $\leftarrow \langle 7,6\rangle$.

6. Figure 3.6
Search:
No change, planned schedule of agents: S1 $\leftarrow \langle\rangle$, S2 $\leftarrow \langle C\rangle$.
Rescue:
R1 has reached 3, R2 has completed 4 and is now allocated to complete 2, R3 has completed 7 and is now allocated to complete 6. Planned schedule of agents: R1 $\leftarrow \langle 3,1\rangle$, R2 $\leftarrow \langle 2\rangle$, R3 $\leftarrow \langle 6\rangle$.

7. Figure 3.7
Search:
S2 has made an observation at C and found one rescue task, task 8. There are no further actions for the search agents.
Rescue:
R1 has completed 3 and is now allocated to complete 1, R2 has reached 2. Planned schedule of agents: R1 $\leftarrow \langle 1\rangle$, R2 $\leftarrow \langle 2\rangle$, R3 $\leftarrow \langle 6,8\rangle$.

8. Figure 3.8
Rescue:
R2 has completed 2 and has no further action assigned to it, R3 has reached 6. Planned schedule of agents: R1 $\leftarrow \langle 1\rangle$, R2 $\leftarrow \langle\rangle$, R3 $\leftarrow \langle 6,8\rangle$.

9. Figure 3.9
Rescue:
R1 has reached 1, R3 has completed 6 and is now allocated to complete 8. Planned schedule of agents: R1 $\leftarrow \langle 1\rangle$, R2 $\leftarrow \langle\rangle$, R3 $\leftarrow \langle 8\rangle$.

10. Figure 3.10
The mission is complete, all 3 search tasks and 8 rescue tasks are completed.
Search:
S1 has completed A, while S2 has completed B and C.
Rescue:
R1 has completed 5, 3 and 1; R2 has completed 4 and 2; R3 has completed 7, 6 and 8.

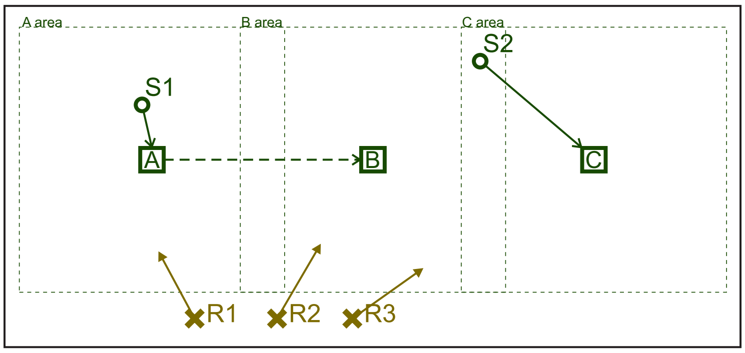


Figure 3.1: Collaboration example initial setting

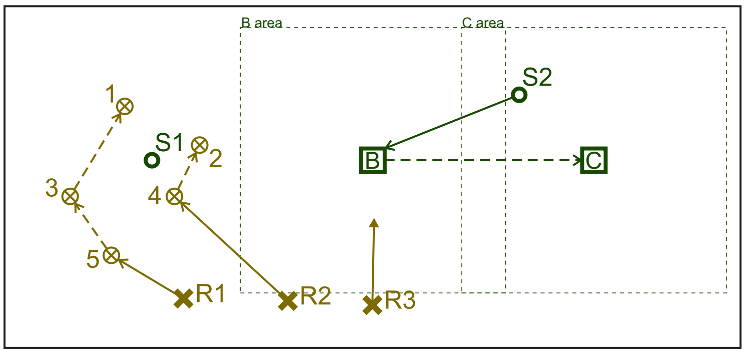


Figure 3.2: Collaboration example second step

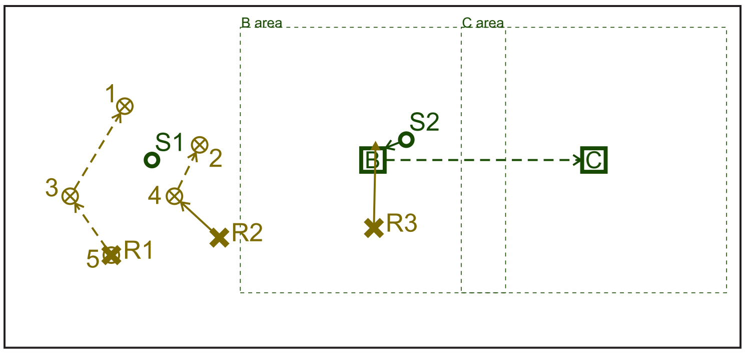


Figure 3.3: Collaboration example third step

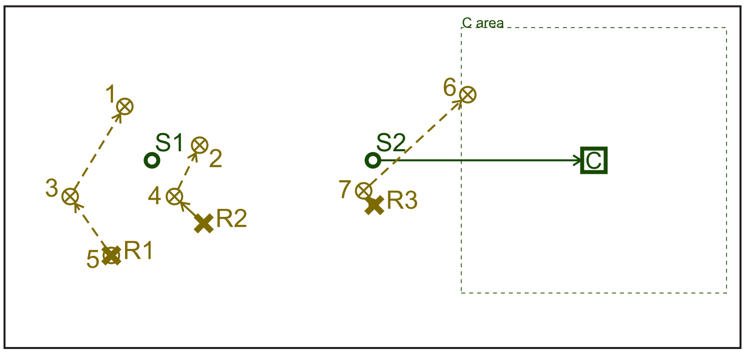


Figure 3.4: Collaboration example fourth step

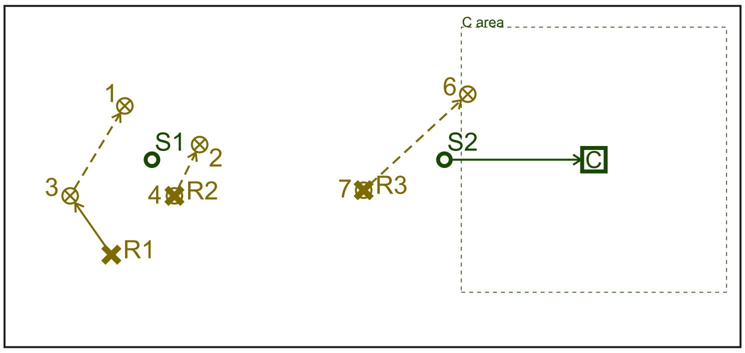


Figure 3.5: Collaboration example fifth step

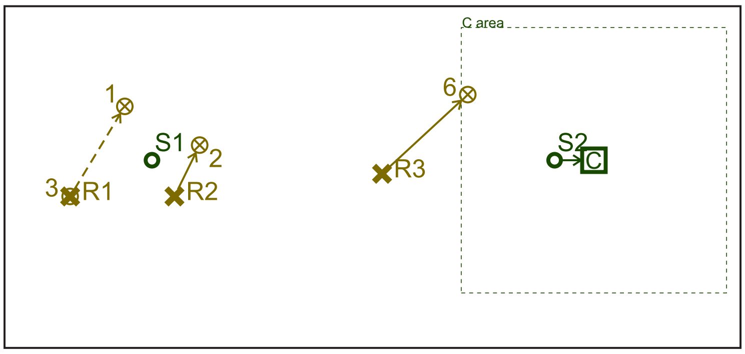


Figure 3.6: Collaboration example sixth step

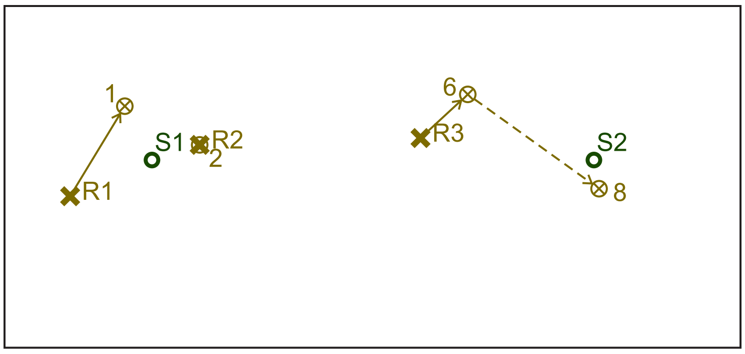


Figure 3.7: Collaboration example seventh step

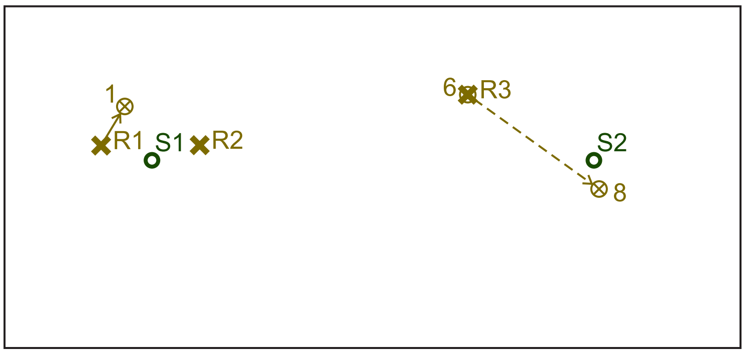


Figure 3.8: Collaboration example eighth step

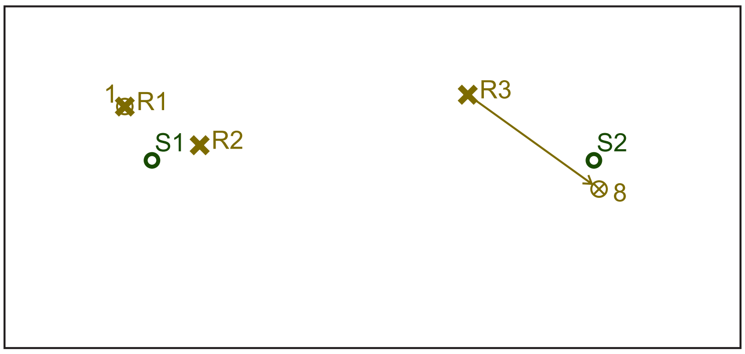


Figure 3.9: Collaboration example ninth step

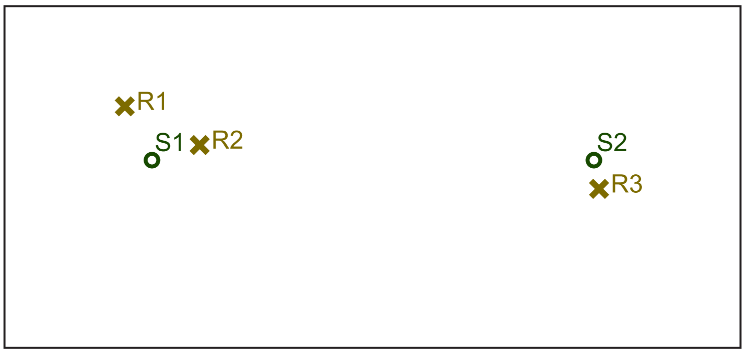


Figure 3.10: Collaboration example final configuration

Our methods are applied to this SAR collaboration problem, but with simple adjustments they could be applied in other problem settings (e.g. more agent roles, more task types, or different representation of uncertainty). Our target is to provide an example solution in this simple setting that can be a basis for different applications in possibly more complex domains (that will be discussed further in Chapter 6).

In the following, we first define the *multi-robot task allocation* (MRTA) problem in a SAR setting, then we propose an extension to it, the *uncertain multi-robot task allocation* (UMRTA) problem introducing uncertainty in the tasks. Finally, we define the *SAR collaboration problem* as a combination of these two problems.

3.1.1 MRTA

In this thesis, we investigate MRTA with single-task robots, single-robot tasks, and time-extended assignment (ST-SR-TA) (Gerkey and Matarić, 2004). This means that each robot can do a single task at once, each task can be done by a single robot, and the robot actions are considered in an extended time horizon (task schedule instead of a single task).

In our context, tasks are divided into two types: *search tasks* (e.g. searching a specific area for victims or assessing collapsed buildings) and *rescue tasks* (e.g. rescue a victim or secure an unsafe structure). By performing these tasks, the robots aim to maximise the number of successful rescues, which must first be discovered by undertaking search tasks. The success rate of rescuing a trapped person exponentially decreases over time as the available data shows (Coburn et al., 1991; Kawata, 1995; Marconi et al., 2013). However, the usual mission time when using UAVs is up to a couple of hours, while the usual survival time is a matter of days, therefore a linear approximation is accurate through the mission time. For this reason, it is modeled as a linearly decreasing utility function[1]:

$$U(\tau) = U_0 - \gamma t(\tau), \tag{3.1}$$

$$\sum_{\tau \in \mathbf{T}} U(\tau) = U_0 |\mathbf{T}| - \sum_{\tau \in \mathbf{T}} \gamma t(\tau), \tag{3.2}$$

where $\mathbf{T}$ is the set of tasks, $t(\tau)$ represents the time of completion of task τ, U_0 is the initial utility of a task, and γ is the utility decrease factor. U_0 and γ are uniform as typically very little is known about tasks before their completion to differentiate them. U_0 and γ are also chosen so that the utility does not go below 0 within the mission time (as explained below, the mission time is much shorter than the usual survival time). Moreover, each of these tasks has a specific location which the robots have to travel to in order to complete it. The time required to travel between the specific tasks can be derived from the motion model of the robot and the traversability of the area. These times can be regarded as necessary setup times to execute a task.

The complexity of the MRTA problem is NP-hard, as the time-extended assignment of a single robot results in a problem very similar to the traveling salesman problem, the orienteering problem (Vansteenwegen et al., 2011) that is NP-hard.

In the following, we define the MRTA problem as a tuple $\langle \mathbf{R}, \mathbf{T} \rangle$, where $\mathbf{R}$ is the set of robots, and $\mathbf{T}$ is the set of tasks. The solution (f) consists of a mapping between the robots and the tasks:

$$f : \mathbf{R} \to \mathbf{T} \cup \{k | k = \varnothing\}, \tag{3.3}$$
$$\text{where } f(a) \neq f(b) \text{ if } f(a) \in \mathbf{T}, a \neq b.$$

The set of tasks are extended with a zero element, $\varnothing$, that resembles no tasks assigned. The mapping must not assign the same task to different robots. However, when using time-extended assignment, a schedule of planned tasks (s) provides the solution rather

[1]Search tasks are not directly associated with a utility, but instead are responsible for discovering rescue tasks (details below).

than a single task:

$$\begin{aligned} s : \mathbf{R} \times \mathbb{N} \to \mathbf{T} \cup \{k | k = \varnothing\}, \\ \text{where } s(a,i) = \varnothing \Rightarrow s(a,i+1) = \varnothing, \\ \text{and } s(a,i) \neq s(b,j) \text{ if } s(a,i) \neq \varnothing, a \neq b. \end{aligned} \tag{3.4}$$

Next, we detail an extension to MRTA with uncertain tasks.

3.1.2 UMRTA

In realistic settings, however, there might not be complete information about the tasks. In this case, the uncertainty about the tasks can be represented as a probability distribution. We define the UMRTA problem as an MRTA problem ($\langle \mathbf{R}, \mathbf{T} \rangle$) with an additional random variable ($\mathcal{T}^+$) representing uncertain tasks[2] : $\langle \mathbf{R}, \mathbf{T}, \mathcal{T}^+ \rangle$.

In this case the optimisation maximises the expected overall utility gain. Accordingly, Equation 3.2 changes as follows:

$$\mathop{\mathbf{E}}_{\mathcal{T}^+} \left[\sum\nolimits_{\tau \in \mathbf{T} \cup \mathcal{T}^+} U(\tau) \right] \tag{3.5}$$

In contrast to the MRTA problem, uncertain tasks cannot be assigned directly because their location is unknown. Therefore we allow the solution to contain a general motion direction for a robot ($\mathbf{D} = [0, 2\pi)$) besides the set of known tasks ($\mathbf{T}$). This allows us to optimise the robot's position given the distribution of uncertain tasks to be closer to these tasks when they are discovered. Of course, these directions need to be reassigned in a timely manner to navigate the robots efficiently. As a result, the solutions will have the following form:

$$\begin{aligned} f : \mathbf{R} \to \mathbf{T} \cup \mathbf{D} \cup \{k | k = \varnothing\}, \\ \text{where } f(a) \neq f(b) \text{ if } f(a) \in \mathbf{T}, a \neq b. \end{aligned} \tag{3.6}$$

Similarly to the MRTA case a time-extended assignment of tasks can provide a schedule of tasks (s) for each robot:

$$\begin{aligned} s : \mathbf{R} \times \mathbb{N} \to \mathbf{T} \cup \{k | k = \varnothing\}, \\ \text{where } s(a,i) = \varnothing \Rightarrow s(a,i+1) = \varnothing, \\ \text{and } s(a,i) \neq s(b,j) \text{ if } s(a,i) \neq \varnothing, a \neq b. \end{aligned} \tag{3.7}$$

Although, this definition is not able to cover a motion direction assigned to a robot when no task is directly assigned to it, when its schedule is empty ($s(a,1) = \varnothing$). Having

[2]Both the properties of the tasks in the set and the cardinality of the set can be uncertain.

described the components of the collaboration problem, we now present the complete heterogeneous robot collaboration problem in a SAR setting.

3.1.3 SAR Collaboration Problem

In this setting, the collaboration breaks into a task allocation problem for the search robots and another one for the rescue robots. The search tasks are known beforehand, while rescue tasks are initially unknown, and will be gradually discovered by search tasks. This means that search can be formulated as a MRTA problem, while rescue is an UMRTA problem. For simplicity, we assume uniform utility and process time for rescue tasks with random 2D locations. Assuming complete spatial randomness, the task locations are the outcome of a non-homogeneous spatial Poisson process (Kingman, 1992) with intensity function λ. During SAR, this distribution frequently changes as new information is introduced about the disaster site or as a region is searched. Within our evaluation we chose a simplistic observation model, meaning that when a region is searched, all tasks within that region are discovered, and the intensity for undiscovered tasks within that region thus drops to zero. This is suitable when the workload for professionals that inspect aerial photos is minimised by each photo being inspected once. The application in Chapter 5 provides a more complex observation model (Section 5.2).

Formally, the search MRTA problem is determined by the set of search robots ($\mathbf{R}_s$) and search tasks ($\mathbf{T}_s$), while the rescue UMRTA problem is determined by the set of rescue robots ($\mathbf{R}_r$) and the known and random set of rescue tasks ($\mathbf{T}_r, \mathcal{T}_r^+$). In this case the random task set, $\mathcal{T}_r^+$, is a Poisson point process ($\mathcal{T}_r^+ \sim \mathit{Poisson}(\lambda)$). Additionally, the connection between the search and the rescue problem can be given by the $\mathit{exploredby} : \mathbf{T}_s \times \mathcal{T}_r^+ \to \{0, 1\}$ function that indicates if a rescue task would be found when a search task is performed (if it falls within the associated area).

Accordingly, the SAR collaboration problem is defined by $\langle \mathbf{R}_s, \mathbf{T}_s, \mathbf{R}_r, \mathbf{T}_r, \mathcal{T}_r^+, \mathit{exploredby} \rangle$ and the solutions can be described by two mapping functions:

$$f_s : \mathbf{R}_s \to \mathbf{T}_s \cup \{k | k = \varnothing\}, \tag{3.8}$$

$$f_r : \mathbf{R}_r \to \mathbf{T}_r \cup \mathbf{D} \cup \{k | k = \varnothing\}, \tag{3.9}$$

$$\text{where } f_s(a) \neq f_s(b) \text{ if } f_s(a) \in \mathbf{T}_s, a \neq b,$$

$$\text{similarly } f_r(a) \neq f_r(b) \text{ if } f_r(a) \in \mathbf{T}_r, a \neq b.$$

Time-extended assignment also results in planned schedules of tasks (s_s, s_r) for each robot as in the previous settings:

$$\begin{aligned} s_s : \mathbf{R}_s \times \mathbb{N} &\to \mathbf{T}_s \cup \{k|k = \varnothing\}, && (3.10)\\ s_r : \mathbf{R}_r \times \mathbb{N} &\to \mathbf{T}_r \cup \{k|k = \varnothing\}, && (3.11)\\ \text{where } s_s(a,i) = \varnothing &\Rightarrow s_s(a,i+1) = \varnothing, \\ \text{similarly } s_r(a,i) = \varnothing &\Rightarrow s_r(a,i+1) = \varnothing, \\ \text{and } s_s(a,i) \neq s_s(b,j) &\text{ if } s_s(a,i) \neq \varnothing, a \neq b, \\ \text{similarly } s_r(a,i) \neq s_r(b,j) &\text{ if } s_r(a,i) \neq \varnothing, a \neq b. \end{aligned}$$

This describes the specific planning problem in the introduced setting. If the problem contains multiple homogeneous robot groups, and their decision making can be formulated into UMRTA (or MRTA) problems, the definition can be formulated in a very similar way, and the planning method can be easily adapted. Next, we are going to detail the planning approaches we applied to solve this problem.

3.2 Planning Approaches

We now introduce the planning methods we compare in this chapter for the above problem. First, we present approaches, where the planning is achieved via decomposition. This includes the state-of-the-art approach in collaborative online planning (decomposition planning with gradient descent), and our novel approach that applies HOP to make decisions under uncertainty (CD). After that, we present our unidirectional planning methods that represent hierarchical planning approaches. Finally, we describe our negotiation-based bidirectional planning approach that creates a joint plan for all agents.

3.2.1 Decomposition Planning Approaches

As concluded in Chapter 2, the most common approach for similar planning problems is decomposition planning. When using decomposition planning, the MRTA problem of search and the UMRTA problem of rescue are solved independently. In this section, we present two approaches applying this principle. The first one uses existing approaches to solve each problem, therefore serves as our benchmark, while the second one uses a novel probabilistic approach to make decisions under uncertainty using HOP.

3.2.1.1 Decomposition Planning with Gradient Descent

After considering several (many of them identical) heuristics to solve the MRTA problem (Gerkey and Matarić, 2004), we chose the shortest adjusted processing time first (SAPT) algorithm (Rabadi et al., 2007), the heuristic that is tailored for tardiness scheduling problems with common due date. In more detail, the SAPT scheduling is often used as a heuristic for similar problems in the scheduling literature, and is optimal for a simpler case with constant travel times for each task and a single agent (Rabadi et al., 2004; Radhakrishnan and Ventura, 2000). Specifically, we define the adjusted processing time as follows:

$$AP\,(\tau_i, [s, \tau_j]) = AP\,(\tau_j, s) + trav(\tau_i, \tau_j) + P(\tau_i). \tag{3.12}$$

$AP\,(\tau, s)$ represents the adjusted processing time of task τ after schedule s ($t(\tau)$ in Equation 3.1, if the schedule is followed by the robot), $[s, \tau]$ stands for a schedule where task τ is inserted at the end of schedule s, $trav(\tau_i, \tau_j)$ is the necessary travel time between task τ_i and τ_j, and $P(\tau)$ is the process time of task τ.

As a standard heuristic planner, the SAPT scheduler is applied to provide a solution for the search MRTA problem $\langle \mathbf{R}_s, \mathbf{T}_s \rangle$, using the intensity map value as a tie-breaker between tasks with the same adjusted processing time[3], and the known part of the rescue problem $\langle \mathbf{R}_r, \mathbf{T}_r \rangle$. This is to prioritise the known tasks in the optimisation, as can be found in the literature for similar problem settings (Rasche et al., 2010; Zivan and Sycara, 2010). In case of idle rescue robots (no known tasks are assigned by the SAPT scheduler), the task intensity map can be used to direct the robots. A widely used and efficient technique for this is gradient descent (Section 2.2.1). In several settings there are large constant intensity areas where the gradient descent is unable to provide a direction. In these cases robots are directed towards central areas in order to minimise the expected distance from upcoming tasks. In detail, rescue robots with no assigned tasks will travel:

- towards the nearest point of the disaster area if outside the affected area,
- otherwise along the gradient of the task intensity,
- or towards the mean of the task distribution (center point) at zero gradient.

This will ensure the minimisation of the distance of the robot from the upcoming task locations.

3.2.1.2 Decomposition Planning with HOP

In this approach, we use the task distribution also for the planning of the rescue. Specifically, HOP (Yoon et al., 2008) (detailed in Section 2.2.3) is applied to solve the UMRTA problem. HOP has been shown to effectively incorporate probabilistic information for

[3]Equal processing time is often when the tasks are placed on a grid. In this case when tasks are placed equal distance from each other on a grid, a similar behaviour to gradient descent is produced.

a similar problem domain for a generic temporal planner (Burns et al., 2012). Unfortunately the introduced temporal planner does not scale well with the size of the problem due to the iteration through the possible resulting states. By contrast, we present a HOP planner that can solve the above presented UMRTA problem with continuous state space. This planner, instead of replanning for all the possible future states, determines the best action using planning for the current state.

The HOP planner incorporates the distribution of unknown task locations using a Monte Carlo simulation. It provides solutions for independent samples of the distribution as if it was a deterministic planning problem (MRTA in this case). As a result, the maximisation criterion (Equation 3.5) is approximated as follows:

$$\mathop{\mathbf{E}}_{\mathcal{T}_r^+}\left[\sum_{\tau\in\mathcal{T}_r} U(\tau)\right] \approx \frac{1}{N}\sum_{i}^{N}\sum_{\tau\in\mathbf{T}_r\cup\mathbf{t}_r^i} U_{sch}(\tau) \tag{3.13}$$

That is, the expected utility is estimated by the average of the utilities ($U_{sch}(\tau)$) determined by the SAPT MRTA scheduler output schedules ($s_{R,i}$) for N samples ($\mathbf{t}_r^i$) of the Poisson process ($\mathcal{T}_r{}^+$). Note that the MRTA scheduler can be replaced with any MRTA solver for a specific application (e.g. an auction-based negotiation combined with RRT path planning (Nanjanath and Gini, 2010) or simulated annealing (Rabadi et al., 2007)).

The maximisation is applied on this utility estimation, and as a result rescue robot actions are determined. This process is detailed in Algorithm 1.

```
Algorithm 1 HOP UMRTA Solver
Require: $\mathbf{T}_r$: set of known rescue tasks
Require: $\mathbf{R}_r$: set of rescue robots
Require: $N$: size of the Monte Carlo simulation
Require: $s_r^i : \mathbf{R}_r \times \mathbb{N} \to \mathbf{T}_r \cup \mathbf{t}_r^i \cup \{k|k=\varnothing\}, i \in \{1..N\}$: hindsight rescue schedules
 1: for all $R \in \mathbf{R}_r$ do
 2:     $\tau^* = \arg\max_{\tau\in\mathbf{T}_r} \mathrm{count}_{i=1}^N \left(\tau = s_r^i(R,1)\right)$    ▷ most common first assigned task
 3:     if $\mathrm{count}_{i=1}^N \left(\tau^* = s_r^i(R,1)\right) > \frac{N}{2}$ then    ▷ if assigned in the majority of cases
 4:         $f_r(R) \leftarrow \tau^*$    ▷ assign task $\tau^*$ to robot $R$
 5:     else
 6:         $d = \mathrm{mean}_{i=1}^N \left[w(s_r^i) * dir\left(R, s_r^i(R,1)\right)\right]$
 7:         $f_r(R) \leftarrow d$    ▷ assign direction $d$ to robot $R$
 8:     end if
 9: end for
Ensure: $f_r : \mathbf{R}_r \to \mathbf{T}_r \cup \mathbf{D} \cup \{k|k=\varnothing\}$: chosen actions for robots
```

Specifically, function $dir(R,\tau)$ calculates the direction that robot R has to take to move towards task τ. In general, for each robot, the aggregation will assign either the most commonly assigned first task in the schedules (τ^*) to the robot, or the weighted average of the direction of the first assigned tasks. In detail, the algorithm iterates over all robots, and finds the most commonly assigned first task ($\tau^* \in \mathbf{T}_r$) in its schedules

(Line 2). If it is assigned as first in the majority of the schedules, the robot's instruction will be to execute task $x_R = \tau^*$, otherwise it will be a general heading direction ($d \in \mathbf{D}$) as in Line 6. This direction is a weighted average of the first assigned tasks' directions, where the weight represents the number of tasks in a schedule ($w(s) = |s|$). This weight tells how many tasks are going to be delayed if the execution of the first task is delayed.

The restriction expressed in Equation 3.9 ($f_r(a) \neq f_r(b)$ if $f_r(a) \in \mathbf{T}_r, a \neq b$) about the rescue solution applies, as a task is only assigned to a robot if in the majority of the samples it is first assigned to the robot (Line 3). This means that the same task cannot be assigned to two different robots as long as the same can be assumed about the hindsight rescue schedules (s_r^i).

This weighted average will provide the *optimal direction of movement* to maximise the utility estimate in Equation 3.13. This, of course, only provides optimality assuming perfect hindsight knowledge of the random task set for each sample ($\mathbf{t}_r^i$). In particular, this means it is only optimal given that the outcome of the random variable is known by the next time step. The proof of this hindsight optimality is given in Appendix A.1.

3.2.2 Unidirectional Collaborative Planning

Unidirectional planning approaches create plans in a sequential manner. Specifically, an independent plan is generated for one of the agent groups, then the other agent group uses this plan as given constraints to create their plan. This can be achieved by choosing either group to create the initial plan. As discussed in Section 2.5, existing approaches use a similar concept where the sequential plan is either done in the same direction as the workflow of tasks (Jing et al., 2009; Lemaire et al., 2004) or the opposite direction (Boumghar and Lacroix, 2011). Although the concept is not novel, these solutions either do not consider uncertainty or only consider it with a limited scope. Against this background, we present two sequential planning algorithms that can fully integrate uncertainty using a custom task set distribution and a HOP technique.

The first step of the planning process, the independent planning is according to the previously detailed decomposition planning approach using HOP (Section 3.2.1.2). While planning with the other agent group happens as follows.

Search-Aware Rescue Planning uses the search plan to introduce a constraint to the execution time of the sampled rescue tasks. Accordingly, the adjusted processing time in Equation 3.12 changes as follows:

$$AP'(\tau, s) = \max\left(AP(\tau, s), cstr(\tau)\right). \tag{3.14}$$

Function $cstr(\tau)$ is the time when task τ is discovered according to the current search plan (temporal constraint). The execution time of each "hindsight task" is delayed until discovered according to the current search plan. The MRTA scheduling problem is still solved using SAPT with the adjusted processing time in Equation 3.14.

Also, the weights (w) in Line 6 of Algorithm 1 need adjustment to ensure optimality. Because the tasks that are delayed due to the temporal constraints do not increase the urgency of a schedule, they should not be counted in the weight. Accordingly, $w(s)$ equals the number of consecutive non-delayed tasks at the beginning of schedule s.

Rescue-Aware Search Planning does not simply minimise the execution time of the search tasks using the plan of the rescue group. The subject of optimisation is to improve rescue, as the utility comes from the rescue tasks. Therefore, the optimisation is applied to minimise the delay introduced on the rescue tasks by the temporal constraints (introduced in Equation 3.14). Due to the nonlinearity of this measure, building an increasing plan by adding tasks to the end of each schedule (as in SAPT scheduling) would not result in a good quality solution. For this reason, a greedy method (Section 2.2.2) sequentially introduces tasks and inserts them to a position of a schedule with minimal cost, as in the MURDOCH negotiation (Gerkey and Mataric, 2002), detailed in Algorithm 2.

Algorithm 2 Search Plan Optimisation

```
Require: $\mathbf{T}_s$: set of search tasks
Require: $\mathbf{I} = \{\mathbf{t}_\tau : \forall \tau \in \mathbf{T}_s\}$: timings from rescue
Require: $\mathbf{R}_s$: set of search robots
 1: $\mathbf{T}' \leftarrow \mathbf{T}_s$: unassigned search tasks
 2: $s_s : \mathbf{R}_s \times \mathbb{N} \to \mathbf{T}_s \cup \{k | k = \varnothing\}$: robot schedules initially empty ($s_s(a,i) = \varnothing$)
 3: while $\mathbf{T}' \neq \emptyset$ do
 4:     $\tau^* = \arg\min_{\forall \tau \in \mathbf{T}'} \min \mathbf{t}_\tau$
 5:     for all $R \in \mathbf{R}_s$ do
 6:         $size_R \leftarrow \text{count}_{i \in \mathbb{N}} (s_s(R,i) \neq \varnothing) + 1$
 7:     end for
 8:     $\langle R^*, i^* \rangle = \arg\min_{\langle R,i \rangle; \forall R \in \mathbf{R}_s, \forall i \in \mathbb{N}, i \leq size_R} \Delta D(s_s, R, \tau^*, i)$      ▷ Minimal delay increase
 9:     $s_s \leftarrow insert(s_s, R^*, \tau^*, i^*)$      ▷ Insert $\tau^*$ to schedule
10:     $\mathbf{T}' \leftarrow \mathbf{T}' \setminus \{\tau^*\}$      ▷ Remove from unassigned tasks
11: end while
Ensure: $\mathbf{S} = \{s_r : \forall r \in \mathbf{R}_s\}$
```

Here, $\mathbf{t}_\tau$ is a set of planned rescue times for sampled rescue tasks discovered by task τ, and the $insert(s, R, \tau, i)$ function inserts task τ into schedule s to the i^{th} location of robot R as detailed in Algorithm 3. Additionally, $D(s)$ estimates the delay caused by search schedule s_s:

$$D(s_s) = \sum_{\tau \in s_s} \sum_{q \in \mathbf{t}_\tau} \max(0, t(\tau, s_s) - q), \tag{3.15}$$

$$\Delta D(s_s, \tau, i) = D(insert(s_s, \tau, i)) - D(s_s). \tag{3.16}$$

Algorithm 3 insert() function

```
Require: $s$: robot schedules
Require: $R$: chosen robot
Require: $\tau$: task to insert
Require: $i$: task location in modified schedule
1: for all $j < i, j \in \mathbb{N}$ do
2:     $s'(R, j) \leftarrow s(R, j)$
3: end for
4: $s'(R, i) \leftarrow \tau$
5: for all $j \geq i, j \in \mathbb{N}$ do
6:     $s'(R, j + 1) \leftarrow s(R, j)$
7: end for
Ensure: $s'$: modified schedules
```

In Equation 3.15, $t(\tau, s)$ denotes the execution time of task τ within schedule s. In brief, Algorithm 2 inserts available tasks – starting with the most urgent ones – into the search robots' schedule using minimal insertion. Specifically, the most urgent task is selected as the one with the soonest rescue time (Line 4). Having selected this task, the best insert location is determined within the current schedules (Line 8). The best position is determined by minimising the delay estimate's increase (Equation 3.16) computed based on the information from the rescue robots' plan ($\mathbf{I} = \{\mathbf{t}_\tau : \forall\tau \in \mathbf{T}\}$).

3.2.3 Negotiation-Based Bidirectional Collaborative Planning

As concluded in Chapter 2, joint planning has not been achieved in an uncertain (CP), multi-role robot setting (CH) using optimisation over a probabilistic model (CD). Against this background, the approach described here will construct a joint plan for the search and rescue robots via a negotiation process. In more detail, the process starts with one of the robot group creating an independent plan and continues with the other group creating a plan using the independent plan as constraints, just as in the unidirectional planning (Section 3.2.2). But the process does not stop here. Now the first robot group creates a new plan using the plans of the other group as constraints, the same way it is described in Section 3.2.2. This process continues on, each iteration one group revises its plans according to the other group's most recent plan, until eventually[4] a joint plan in created for both groups. This way, the solution quality can be further improved by taking the relations between the search and rescue activities into account in both directions. As a result, the search robots are going to be more likely to visit areas that are visited shortly after by rescue agents, and also rescue robots are going to approach valuable locations that are planned to be searched sooner.

[4]The iteration length within 5 and 9 does not change the overall result significantly, so we chose 5 iterations in the evaluation.

3.3 Empirical Evaluation

The evaluation resembles the scenario introduced in Section 1.2. More specifically, it captures the collaboration of fixed-wing and rotary-wing UAVs in the described setting, therefore the actions of the bronze commanders are not considered. In more detail, fixed-wings search the area by taking aerial images. These images are analysed (by either a professional or computer vision algorithms (Doherty and Rudol, 2007; Andriluka et al., 2010)) to point out sights of interest such as possible victim locations, dangerous buildings or critical supplies. The sights of interest are visited by rotary-wing UAVs to provide detailed, close-up video feeds to first responders. This procedure allows first responders to assess an area and be able to locate some high priority tasks before accessing the disaster site.

The underlying planning problem is the SAR collaboration problem described in Section 3.1.3. In particular, fixed-wing UAVs take the role of the search robots, and rotary-wing UAVs take the role of rescue robots.

At different disaster sites, the settings can be very different depending on the cause of the disaster (e.g. earthquake, extreme weather or industrial disaster), the impact of the disaster or the population density of the area. For this reason, our experiments include two very different settings from two disasters mentioned in Section 1.1, namely the Ajka Alumina Plant industrial spill (2010) (Figure 3.12a) and a sector search after the Haiti earthquake (2010) (Figure 3.12b). The former has a sparse effect spreading along a large region, while the latter represents a larger impact disaster in a smaller area. The area size and the density of tasks will determine if rescue agents spend more time moving between rescue locations or performing rescues. In the following we detail the settings and the numerical results of the conducted experiment.

3.3.1 Experimental Setup

It is difficult to tell the level of information available when a SAR team starts operating after a disaster, even though the level of available information can be a crucial factor affecting an algorithm's performance. For this reason, we evaluated the performance of the planning approaches in three settings, namely, the two extremes (i.e., lowest and highest possible levels of information) and a less extreme and more plausible setting (in which we assume disaster resilience).

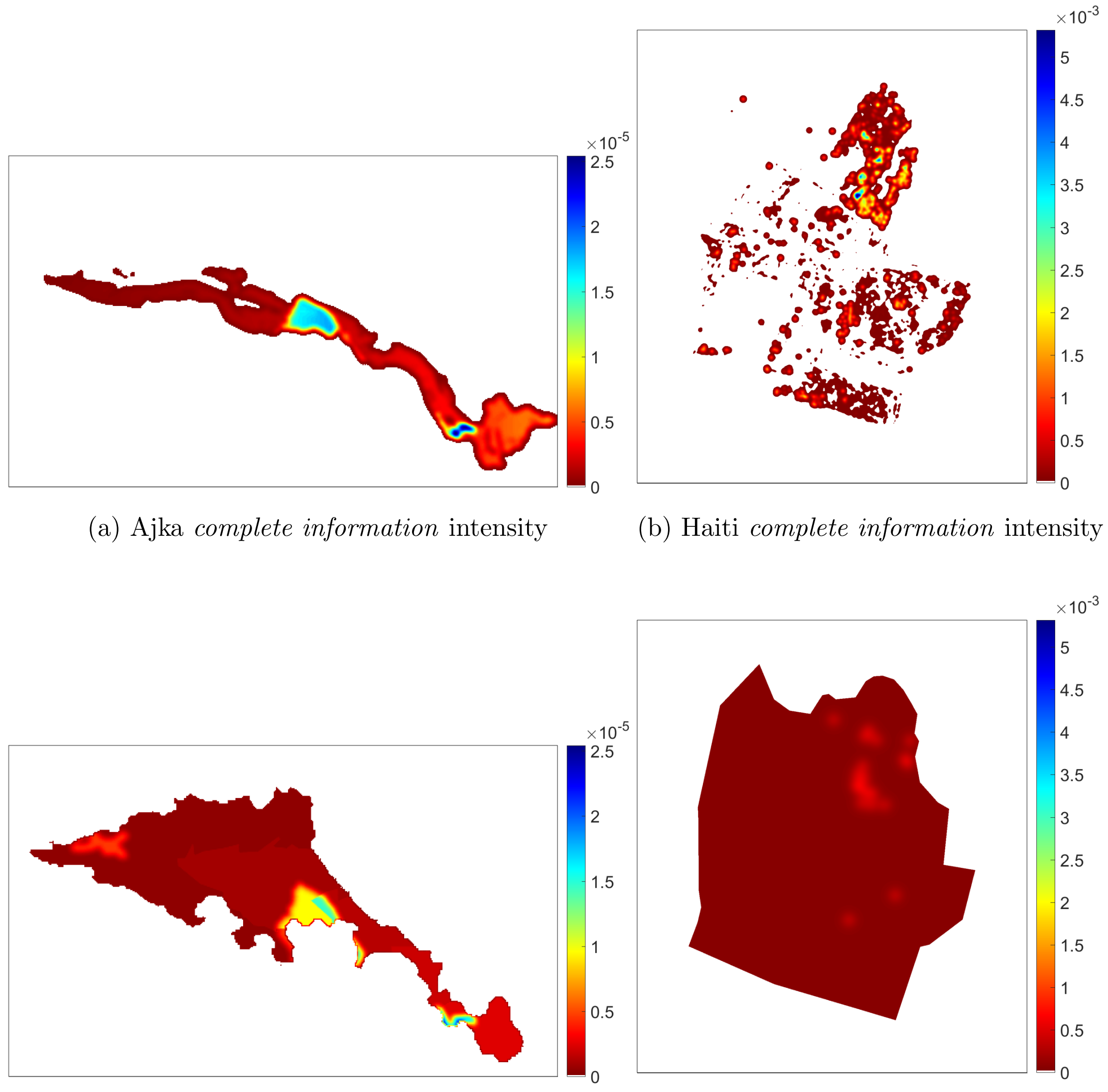
- *Complete information*: represents an ideal setting when the SAR team knows everything about the disaster's impact. The belief distribution (λ) is identical to the distribution from which the victim locations are drawn. This distribution is generated using detailed disaster assessment data acquired weeks or months after the disaster.

- *Resilient setting*: represents a realistic setting with disaster resilience planning in place (Birkmann, 2013) that can provide some information about the disaster's impact as the SAR team arrives to the disaster site. The belief distribution (λ) is a simplified version of the distribution from which the victim locations are drawn. This distribution is generated using information that can be provided immediately after a disaster happens (e.g. physical simulations, high resolution population density data or early victim reports from the disaster site).
- *Zero information*: represents a worst-case setting when no information is available as the SAR team arrives to the disaster site. The belief distribution (λ) is uniform inside the SAR operation area.

In more detail, the belief distribution (λ in Section 3.1.3) that represents the prior information about the disaster impact is constructed the following way:

- *Complete information*
  - *Ajka* scenario: A combination of the built-up regions and the flooded area (with decreasing intensity with the distance from the source) covering 5.1 km^2 taken from Ambrus (2011). (Figure 3.11a)
  - *Haiti* scenario: Victim intensity in a search sector of Port au Prince based on a building damage assessment with intensities according to Table 3.2 covering 0.106 km^2 taken from UNOSAT (2010). (Figure 3.11b)
- *Resilient setting*
  - *Ajka* scenario: A combination of the built-up regions and the results of a spill simulation for a similar spill incident covering a larger possible area (12.7 km^2) taken from Police of Hungary (2010). (Figure 3.11c)
  - *Haiti* scenario: Uniform inside the search sector (0.875 km^2) with higher intensities around 20 simulated report locations (drawn from the *Ground Truth* intensity). (Figure 3.11d)
- *Zero information*
  - *Ajka* scenario: A uniform intensity (2.6 10^{-7} task/m^2) over a large rectangular area (78.5 km^2, edges marked with the frames of Figures 3.11a and 3.11c) above the possible area of the industrial spill.
  - *Haiti* scenario: A uniform intensity (6.9 10^{-5} task/m^2) inside a search sector (0.875 km^2) in Port au Prince.

In order to quantify the amount of information conveyed by the different information settings, we used the Kullback-Leibler (KL) divergence (Kullback and Leibler, 1951) of the distributions of the information settings from the exact distribution from which the



(a) Ajka *complete information* intensity

(b) Haiti *complete information* intensity

(c) Ajka *resilient setting* intensity

(d) Haiti *resilient setting* intensity

Figure 3.11: Intensity maps (units are task/m^2)

task locations are drawn. It is calculated for the 2D spatial Poisson processes using to the following formula (p and q are the intensity functions of process P and Q respectively):

$$D_{KL}(P||Q) = \iint_A p(x,y) \log \frac{p(x,y)}{q(x,y)} \mathrm{d}x\mathrm{d}y. \tag{3.17}$$

This measure is able to tell the divergence of an approximate distribution from the original as it quantifies the information loss from sampling the approximate distribution instead of the original distribution. Larger values mean more information loss from the original distribution, therefore we expect an intelligent optimisation to perform worse when given a belief distribution with a higher divergence. This is because the optimisation is performed given the approximate distribution and the larger the divergence from the original distribution, the less likely the optimisation will be efficient on the original

Information setting (Q)	$D_{KL}(P\|\|Q)$
Ajka *Zero information*	3.157
Ajka *Resilient setting*	0.234
Ajka *Complete information*	0
Haiti *Zero information*	2.287
Haiti *Resilient setting*	1.640
Haiti *Complete information*	0

Table 3.1: Kullback-Leibler divergence of the estimate distributions from the *Ground Truth* distributions of the same setting

Tag	Description	**E**[#rescues]
Grade 1	Negligible to slight damage	0.002
Grade 2	Moderate damage	0.004
Grade 3	Substantial to heavy damage	0.01
Grade 4	Very heavy damage	0.04
Grade 5	Destruction	0.125

Table 3.2: Haiti assessment (UNOSAT, 2010) damage levels and expected number of rescues[5] at each tag location

problem. The minimum value of this measure is $D_{KL}(P||Q) = 0 \iff P = Q$ (that is the case for the *Complete information* setting). Consequently, the KL divergence is zero for the *Complete information*, a high value for the *Zero information*, and a lower value for the *Resilient setting*. The values of this measure for the specific distributions can be found in Table 3.1.

The performance of the different planning approaches is compared in simulation for 128 different disaster outcomes for both scenarios. Each outcome is an independent sample from a Poisson process with the *complete information* distribution introduced above. Each approach is tested with the three different levels of available information independently.

The experiments, simulate the two groups of robots (search and rescue) performing collaborative SAR. Table 3.3 details the chosen parameters for these simulations[6]. The parameters are chosen to match realistic settings as much as possible. The parameters for the two scenarios differ due to the differences in the two disaster environments. Specifically, the Haiti sector is a densely populated urban area where the large number of features require sub-cm resolution, while large fields covered with red mud near Ajka require lower resolution imagery for victim search. The search imagery area size was chosen according to these measures to include a similar number of features. Consequently, the simulation time step is chosen in a way that the search robots need 4-6 time steps to travel between two search locations. This allows a human supervisor to intervene between actions, and also to interact with a reasonably responsive decision making system

[5]Expected number of points in a Poisson point process equals the integral of the intensity function within an area.

[6]We got broadly similar patterns of results for scenarios where other values were explored.

Parameter	Search robots	Rescue robots
Robot speed	25 m/s	10 m/s
Task length	0 s	30 s

(a) Robot parameters

Parameter	Ajka scenario	Haiti scenario
Search grid spacing	440 m	105 m
Search imagery area	530×530 m	125×125 m
Simulation time step	3 s	1 s
#samples (N)	100	100
#search robots	2	2
#rescue robots	4	8
#search tasks	400, 120, 79	95, 95, 87
#rescue tasks	20 (avg.)	61 (avg.)

(b) Scenario parameters

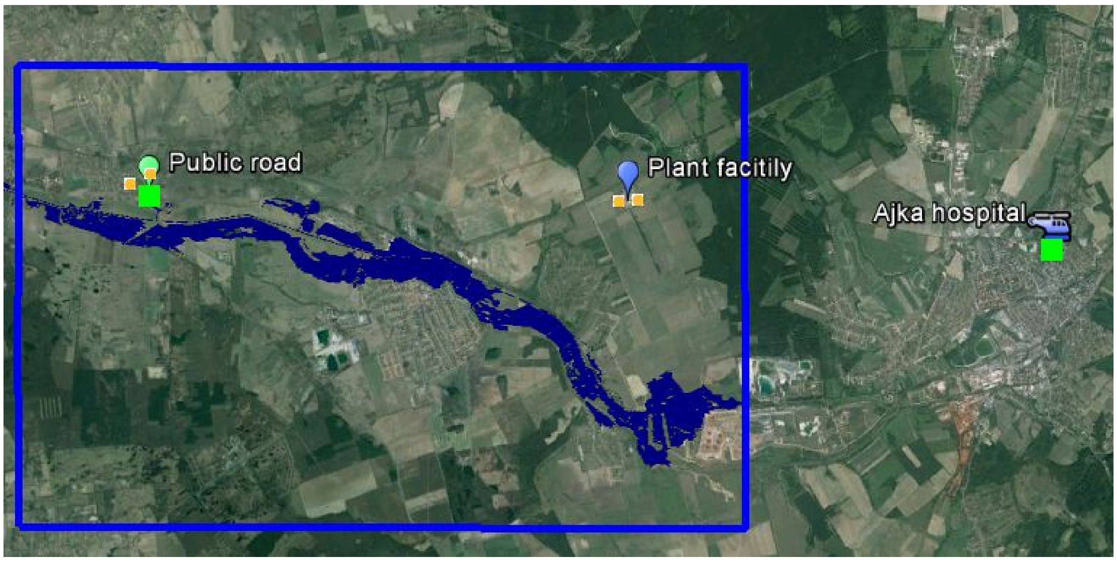
Table 3.3: Simulation parameters

(Nielsen, 1993). However, varying the simulation time step did not change the result significantly. Search tasks are located on grid points where the intensity is greater than zero ($\lambda > 0$) for the corresponding area.

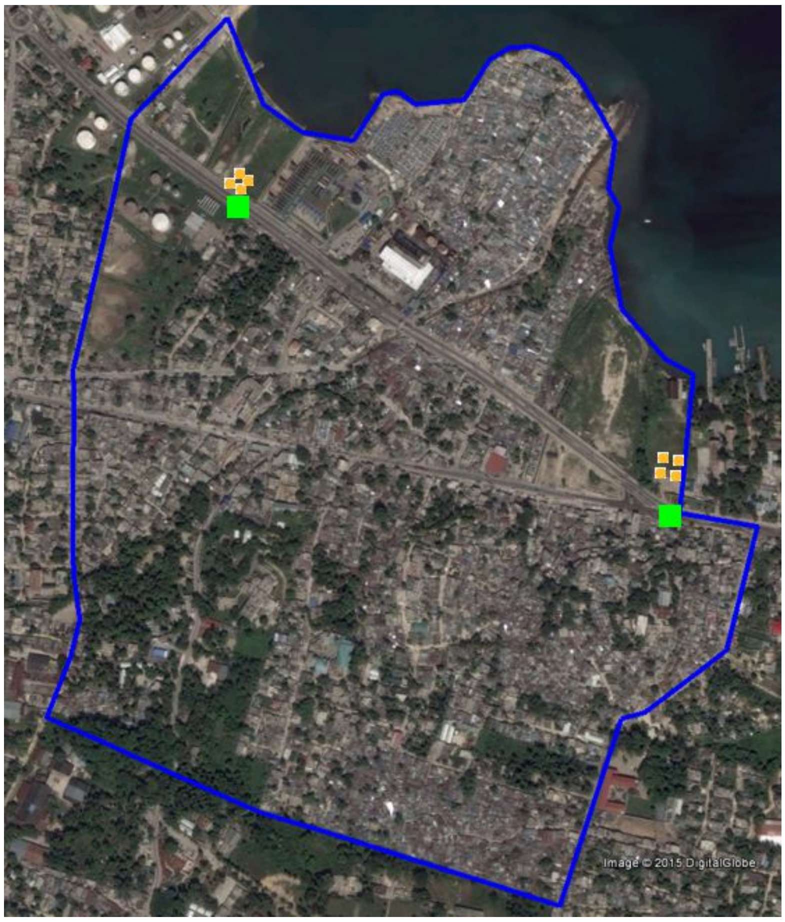
The start locations of the UAVs can be seen in Figure 3.12b and in Figure 3.12a over the satellite imagery of the area. The edge of the SAR operation area (the boundaries of the zero information belief distribution) is marked with blue, yellow markers show rotary-wing start locations, and green rectangles mark the start location of fixed wing UAVs. The fixed-wing start locations are placed on possible take-off locations such as nearby airports and intact straight roads, while rotary-wing start locations are chosen at rubble-free locations where the infrastructure can be set up for controlling them.

Each simulation instance used a single core of an Intel Xeon E5-2670 processor. The computation time is measured for the bidirectional approach to evaluate its computational feasibility. More information about the simulation framework can be found in Appendix B.1.1.

In the following, the results of the conducted experiments are introduced. The statistical significance is tested by comparing the relative performance against our approach for each run that used the same rescue locations by a one-sample t-test. Specifically, we use the average time of the rescue operations as a performance indicator. This gives a direct comparison of the overall utility gained by the different approaches according to the utility definition (Equation 3.2). The lower this metric, the higher the SAR performance.

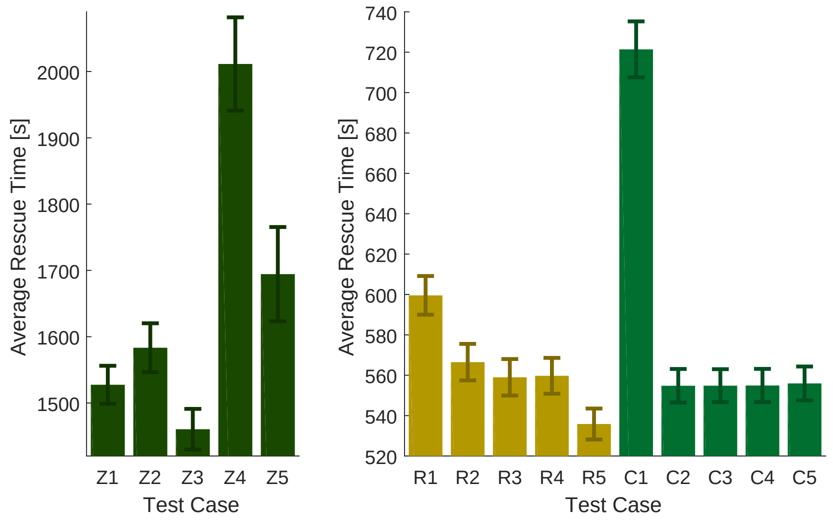


(a) UAV start locations for the Ajka scenario

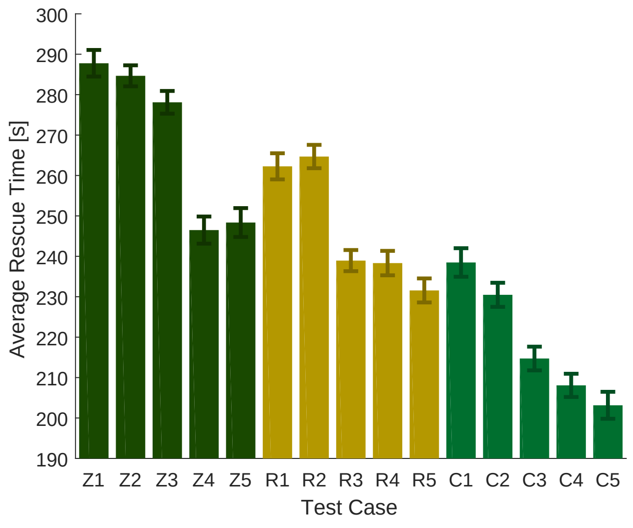


(b) UAV start locations for the Haiti scenario

Figure 3.12: UAV start locations over satellite imagery



(a) Ajka scenario performance



(b) Haiti scenario performance

Figure 3.13: Average rescue time comparison with 95% confidence intervals.

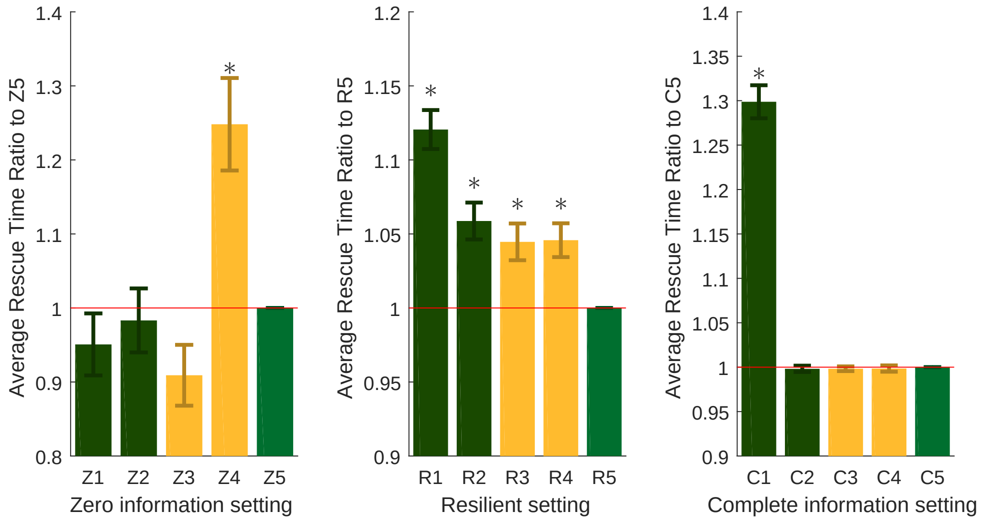
3.3.2 Experimental Results

In this section the performance and the computation time is evaluated for the previously detailed approaches in the above settings. Within the comparison (Figures 3.13 and 3.14), there is a two character code identifying each test case, assembled by the first letter of the information representation and the number of the planning approach. The notation of the setting depending on available information is the following:

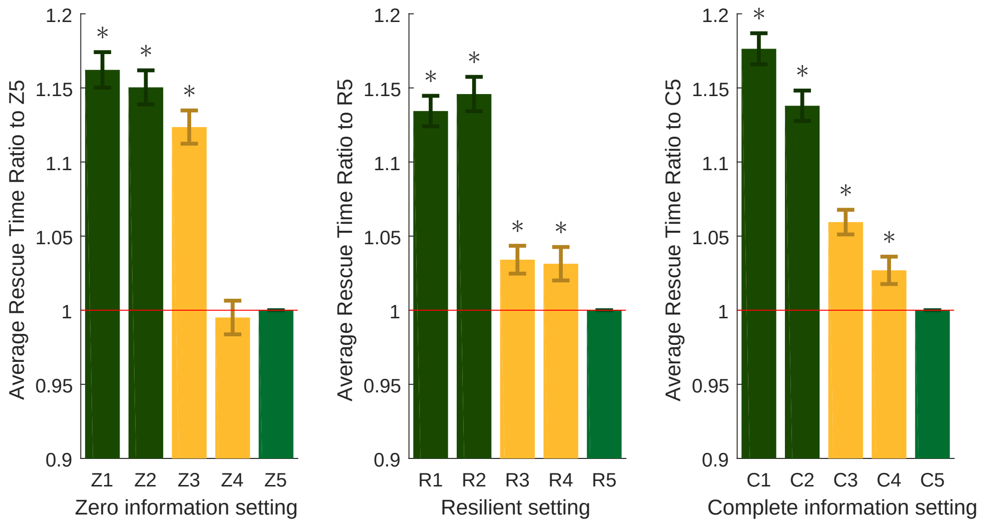
Zx: *Zero information* setting,

Rx: *Resilient setting*,

Cx: *Complete information* setting.



(a) Ajka relative performance



(b) Haiti relative performance

Figure 3.14: Relative performance comparison with 95% confidence intervals (stars mark approaches that perform significantly worse than the bidirectional approach).

Likewise, the collaboration approaches are:

x1: *Decomposition planning* with gradient descent,

x2: *Decomposition planning* with HOP,

x3: *Unidirectional collaborative planning* along the workflow structure,

x4: *Unidirectional collaborative planning* against the workflow structure,

x5: *Negotiation-based bidirectional collaborative planning.*

For instance, “C3” stands for the complete information setting and unidirectional collaborative planning along the workflow structure approach, or “R2” stands for resilient setting and decomposition planning with HOP approach.

The **Overall Performance Comparison** can be seen in Figure 3.13, while the relative performance to the bidirectional approach can be seen in Figure 3.14. The performance is compared by the average time that rescue tasks are completed aligned with the utility definition in Equation 3.1, the lower this measure the better the performance. Firstly, an improvement of 15-30% in the average rescue time can be observed between *decomposition planning with gradient descent* (the state-of-the-art for complex collaborative planning under uncertainty) and our *negotiation-based bidirectional collaborative planning* approach for the cases with sufficient available information. This shows that long-term planning under uncertainty is able to make strategic decisions to divide the disaster area between the agents, especially when these plans are jointly optimised for all robots. The *negotiation-based bidirectional collaborative planning* approach (Z5, R5, and C5) performs significantly better than the benchmark approach for the same problem setting (Z1, R1, and C1) in almost all cases. In the realistic resilient setting, the rescues are performed 12% earlier for the Ajka scenario, and 13% earlier for the Haiti scenario compared to the benchmark. However, the joint planning approach does not perform well for the *zero information* setting in the Ajka scenario (Z5 in Figure 3.14a). This setting conveys an outstandingly low amount of information (the highest value in Table 3.1) as the distribution covers a 15 times larger area than that which is actually affected by the disaster. In such cases with a large uniform rectangular area, the solution for search becomes trivial as the lawnmower pattern (the result of SAPT scheduling) is optimal (Coles et al., 2010).

In terms of collaborative planning, the bidirectional planning outperforms both unidirectional planning approaches, except for the zero information settings, and the Ajka complete information setting (where there is no benefit from collaborative planning). We can also see that in the cases with the highest information about the actual rescue task distribution (Ajka resilient setting and both complete information settings – the lowest values in Table 3.1) HOP significantly improves the efficiency of decomposition planning. This shows, that using HOP for UMRTA planning is only beneficial when there is sufficient information available about the uncertain process. However, from the overall performance comparison in the Ajka setting (Figure 3.13a) we can see that the performance of the gradient descent approach highly relies on minor details of the belief distribution. Specifically, while the complete information and the resilient setting convey a very similar amount of information (the Kullback-Leibler divergence is only 0.234), the gradient descent approach performs rescues 20% later in the complete information setting (C1) than in the resilient setting (R1).

The **Computation Time** of the bidirectional planning for different numbers of rescue tasks can be seen in Figure 3.15. The majority of the computation time is because of the rescue-aware search planning[7] (Algorithm 2 in Section 3.2.2). This algorithm's

[7]The computation time spent by search-aware rescue planning in the Ajka setting is below 1 ms, and the complexity of the SAPT heuristic is square in the number of tasks ($\mathcal{O}(|\mathbf{T}_r|^2)$) (Rabadi et al., 2007).

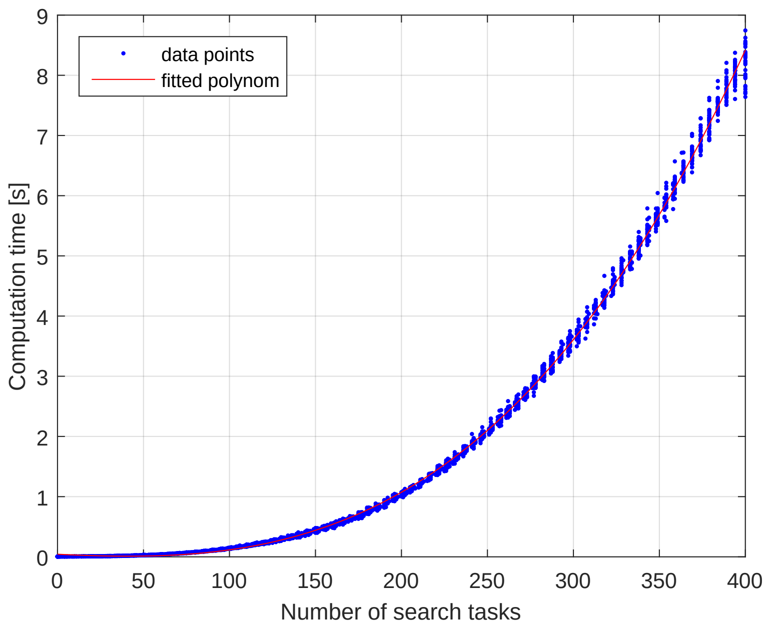


Figure 3.15: Computation time against the number of rescue tasks

complexity is cubic in the number of search tasks ($\mathcal{O}(|\mathbf{T}_s|^3)$), as it iterates through the search tasks (Algorithm 2, line 3), then finding the best insertion (Algorithm 2, line 8) requires to iterate through the existing schedules, and each time through the rest of the schedule. This tendency can also be seen in Figure 3.15, as the fitting curve is a third order polynomial. This means that the number of search tasks needs to be kept below 200–300 in order to result in a responsive system (Nielsen, 1993) using the bidirectional collaborative online planning. This restricts the scalability (RS) of this solution.

3.4 Summary

In this chapter, the SAR collaboration problem is defined, a planning problem that fulfills the criteria listed in Section 1.3. As highlighted in Section 1.4, an algorithm is introduced for solving the uncertain multi-robot task allocation (UMRTA) problem to control robots in a continuous space given an arbitrary task distribution. The proposed method provides a possibility for creating a long-term collaborative plan in an uncertain setting (CH and CD). Making use of this ability, we proposed an approach for the SAR collaboration problem, that is the first to provide bidirectional collaborative planning. As a result, a *joint* plan is produced for both search and rescue robots. The planner provides a flexible (RF) and computationally efficient (RR) solution. The SAR collaboration problem has real-life implications in multi-UAV planning in a disaster response setting. We describe an aerial imagery based application for the SAR collaboration problem, and evaluate our algorithm against other approaches from the literature (see

Section 2.5). The introduced planner[8] rescues 12-13% earlier compared to the state of the art approaches in realistic settings. Moreover, computation time is maintained on an acceptable level for settings with less than 200–300 search tasks (RR).

However, this approach does not provide a scalable (RS) solution, as it cannot provide a real-time solution when there is a large number of search tasks. In order to solve this, we propose an anytime algorithm for the collaborative SAR planning problem in the following chapter. An anytime approach can be stopped whenever the computational budget is reached, therefore provides a real-time (RR) plan in any setting (RS). Similarly to the algorithm presented in this chapter, the solution builds on long-term planning under uncertainty using HOP.

[8]The following example videos compare R1, R2 and R5 settings called as "State of The Art" (left), "Hindsight Planning" (middle) and "Joint Planning" (right) respectively:
`https://www.dropbox.com/s/rrpzsxkk7fv4wrv/combinedAjka.mkv?dl=0`,
`https://www.dropbox.com/s/8gv1euzclgym3wi/combinedHaiti.mkv?dl=0`.

Chapter 4

Combined Monte Carlo Tree Search for Collaborative Planning

As concluded in the previous chapter, joint planning in an uncertain collaborative setting such as the SAR collaboration problem is beneficial, as it increases the efficiency in various settings. However, collaborative search and rescue can become a computationally intensive problem, because capturing uncertainty of a large decision space makes the optimisation complex, and can easily face scalability issues (RS). Given this, it is important to use the computational resources effectively and adaptively for different degrees of complexity or computational budgets. The effectiveness can be assured by targeting a continuous search for a better plan towards the most promising areas. The adaptivity to a specific computational budget can be achieved by using a solution that runs in an anytime manner, meaning that it can be stopped at any time the computational limit has been reached. This results in near-optimal solutions for simple problems, but a good quality solution is found in more complex cases (RS) whilst keeping the real-time limit (RR).

In this vein, Monte Carlo tree search (MCTS, introduced in Section 2.2.4) is a powerful approach for complex decision making and it provides a solution that covers both of the above criteria. Specifically, it provides an anytime algorithm that increasingly extends the search towards the areas that have given the best solutions so far. For these reasons, MCTS is an appropriate tool for complex heterogeneous collaboration under uncertainty and is the basis of the solution defined in this chapter.

This chapter explains the application of MCTS with HOP (Sections 2.2.3 and 2.2.4) for collaborative planning under uncertainty. In particular, it is applied to solve the SAR collaboration problem introduced in Section 3.1.3. The MCTS approach follows the same four iterative steps as detailed in Section 2.2.4. However, there are several changes to the tree structure and the individual steps in order to use MCTS for collaborative planning compared to its typical domain of gameplay. Firstly, the modifications of the classic MCTS tree structure to fit our model are explained in Section 4.1. After that, the heuristics used in the node selection, including a novel heuristic based on the value of information, is detailed in Section 4.2. Section 4.3 details the heuristics used in the expansion process. This is followed by an empirical evaluation of the performance of the collaboration using MCTS compared to the approaches in Chapter 3 (Section 4.4). Finally, the work is summarised in Section 4.5.

4.1 Tree Structure

As explained in Section 3.1.3, the SAR collaboration problem consists of two coupled planning problems: the planning of search activities and the planning of rescue actions. The coupling between these two planning problems lies within rescue locations being found during the search process. The search tree (an example is shown in Figure 4.1) captures this coupling.

There are several differences in the tree structure compared to the typical domain of MCTS (gameplay). During gameplay, the player and one or more opponents[1] take actions in turns. The set of actions are limited and the intermediate states of the game provide feedback after each action. In collaborative planning, on the other hand, there is no opponent, but the complex action plans of the agents are inspected over the possible outcomes of the uncertain process. In our application, the levels of the tree represent plans of the partial planning problems of an agent group in a specific role or the different outcomes of a random process, rather than the sequential turns of players in classical gameplay MCTS. This will result in a tree of a limited depth, but with large branching, as the partial planning problems are usually complex (a high number of solutions exist) and the random process can be continuous. Also, the different nodes of the search tree do not represent game states, but represent the original planning problem with constraints (e.g. constraints from ancestor solutions or a specific outcome of a random process).

This novel tree structure provides a possibility to dynamically explore the uncertainty the same way as the action space, as opposed to the fixed-width HOP in other approaches combining MCTS and HOP in solitaire gameplay (Bjarnason et al., 2009) (discussed in Section 2.2.4). The tree for the SAR collaboration problem (Figure 4.1) consists of the following four levels:

1. **Search problem**: describes an MRTA problem for the search ($\langle \mathbf{R}_s, \mathbf{T}_s \rangle$), the root of the search tree.
2. **Search solution**: provides a solution schedule to the *search problem* ($s_s : \mathbf{R}_s \times \mathbb{N} \to \mathbf{T}_s \cup \{k | k = \varnothing\}$).

[1]There is a special domain for single-player games that provide a solution much closer to planning, a relevant work (Bjarnason et al., 2009) in combining MCTS and HOP is applied to a single-player game.

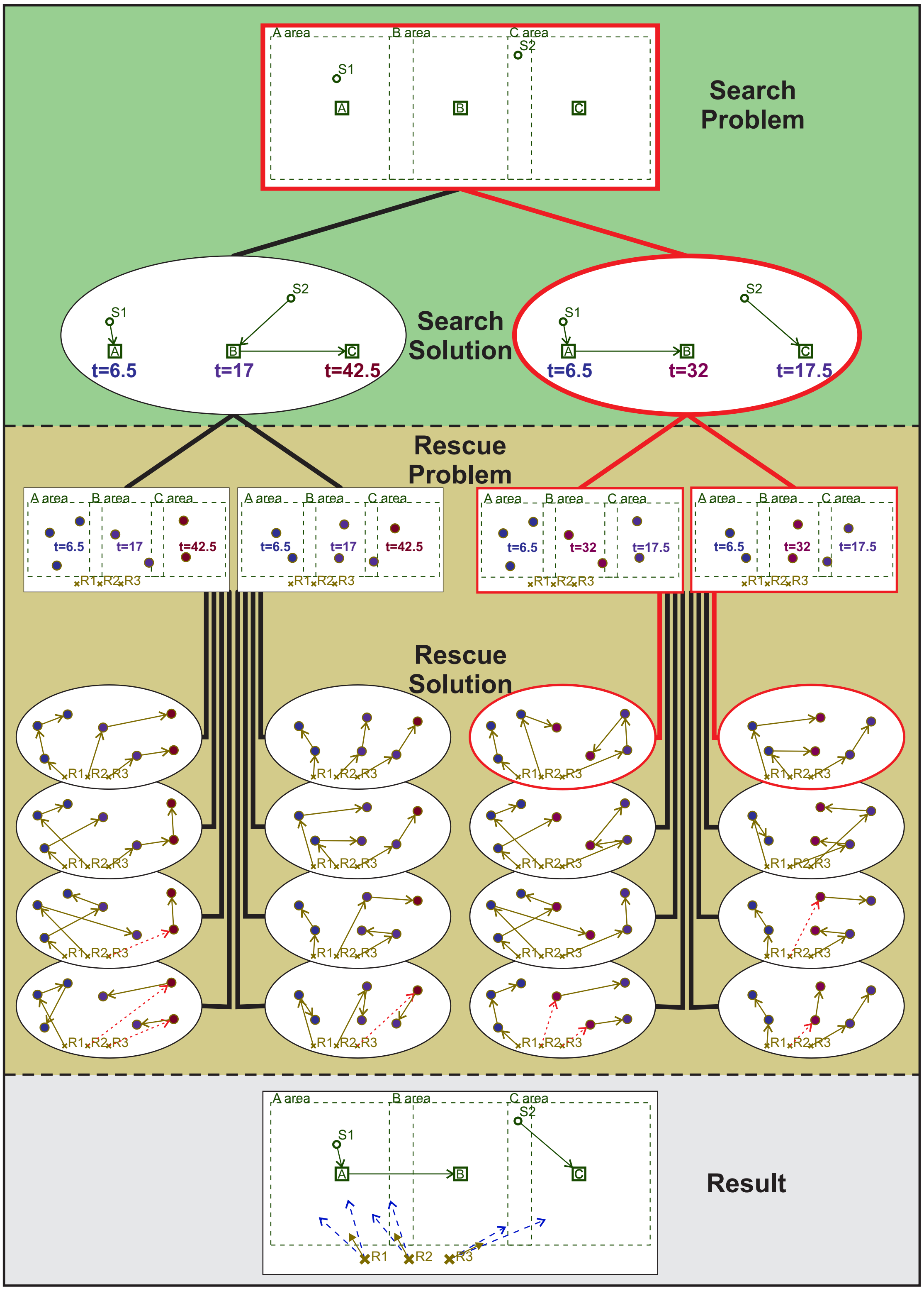


Figure 4.1: Tree structure example of the collaborative search and rescue planning.

3. **Rescue problem**: represents a sample in the HOP, describes an MRTA problem for the rescue with random task locations assumed to be known in hindsight. Temporal constraints ($cstr : \mathbb{T}_r \to \mathbb{R}^+$, where $\mathbb{T}_r$ is the set of all possible rescue tasks) are introduced from the parent *search solution* as explained in Section 3.2.2 ($\langle \mathbf{R}_s, \mathbf{T}_r \cup \mathbf{t}_r^i, cstr \rangle$).

4. **Rescue solution**: provides a solution schedule to its parent *rescue problem* ($s_r : \mathbf{R}_r \times \mathbb{N} \to \mathbf{T}_r \cup \mathbf{t}_r^i \cup \{k | k = \varnothing\}$).

Figure 4.1 provides an example for the tree structure and the selection process through the initial problem of the example in Section 3.1. The notation is the same, green circles are search robots, green rectangles are search tasks, golden crosses are rescue robots, while golden circles are rescue tasks. The temporal constraints are marked by the colours of the rescue task marker filling. The rescue solutions are sorted by their utility, the topmost solutions have the highest utility. Allocations are marked with green or golden arrows, red dashed arrows mark allocations delayed by the temporal constraints (non-critical allocations). These cause delays in the whole schedule that has a strong impact on the utility (the best *rescue solutions* for the left *search solution* provide the worst *rescue solutions* for the right *search solution*). There are three possible expansions in this limited depth tree:

1. Expansion of the *search problem* provides alternative *search solutions*. This explores the possible actions of the search robots.

2. Expansion of a *search solution* results in *rescue problems*. The new rescue problems have different rescue task locations according to a new sample of the uncertain rescue tasks ($\mathcal{T}_r^+$). This explores the uncertainty of the rescue task locations, providing a more accurate estimate by increasing the number of HOP samples.

3. Expansion of a *rescue problem* provides alternative *rescue solutions*. This explores the possible actions of the rescue robots.

The nodes selected during the *winning action selection* process are marked with red. This includes the selection of the best solution child and all problem children. This way a single search solution and multiple rescue solutions are selected, that are aggregated according to Algorithm 1. The resulting actions can be seen at the bottom of Figure 4.1, the blue dashed arrows represent the vectors taking part in the averaging process (Line 6).

Accordingly to this new tree structure, the adjustments of the main steps of MCTS (as in Section 2.2.4) are explained in the following sections.

4.1.1 Selection

The first step in each iteration of MCTS is *selection*. It decides which node of the tree will be expanded.

Due to the large branching and short tree-depth explained above, conventional MCTS tree growth is not applicable in this case, as it is not feasible to expand the higher level nodes with all possible children before expanding a child node. This means that the expansion at different levels needs to be compared to find the most beneficial expansion step. On the other hand, due to the limited tree depth, this needs to be considered on a limited number of levels only. Because of the higher number of children nodes of each node, there is more information available about the children node population than in classical MCTS. This makes more elaborate statistics possible about the children nodes than the most commonly used UCB method (Kocsis and Szepesvári, 2006). Such statistics are also necessary in order to compare expansions at different levels. To achieve this, the gain from a specific expansion needs to be placed into the scope of the collaborative planning problem. This can be achieved by, for example, expectation statistics or value of information approaches besides UCB. The actual applied heuristics for selection are detailed in Section 4.2.

4.1.2 Expansion

After the selection process, multiple children are added to the selected node during *expansion*. In this way the advanced, more computationally complex statistics can be computed less frequently in order to save computation time.

The expansion is a straightforward process. New nodes are generated by providing alternative solutions to an MRTA problem or making new samples for the HOP.

The expansion of a *problem node* means providing solutions for the MRTA problem represented by the node. The initial solution is the same as the one applied in Section 3.2.1.2, namely the SAPT schedule. Specifically, when a *rescue problem* is expanded, the temporal constraints from the *search solution* are taken into account as in the search-aware rescue planning in Section 3.2.2. Further children are generated similarly to the SAPT scheduling as a result of a random process, to be able to produce alternative solutions. More detail about the chosen random process can be found in Section 4.3. During this process, individual task allocations are a result of a weighted random choice depending on the adjusted processing time, instead of choosing the earliest adjusted processing time tasks each step as in SAPT scheduling. These random solutions are enhanced according to Algorithm 7 (Appendix A.2), to ignore obviously suboptimal schedules. The resulting population of random solutions provide a variety of different plans that have a good chance to outperform the initial SAPT schedule (as it can be seen in Figures 4.8b and 4.9b). The more alternative solutions are created this way, the higher the expected utility of the best result. This ensures the anytime behaviour of the planner.

The expansion of *solution nodes* is achieved by drawing a sample from the task distribution that results in a MRTA problem. This process extends the number of HOP samples processed for a specific *solution* in order to explore its uncertain outcome. To decrease the sampling effort and memory usage, the n^{th} child of each *solution node* will use the same HOP MRTA problem sample. In this way, the sampling noise of the utility estimate is also decreased for smaller number of samples, because the different subtrees will compute their utilities based on the same (or very similar) problem set.

4.1.3 Simulation

The *simulation* phase to determine the initial utility of a child becomes simpler due to the limited tree depth. The typical solution for this is to apply a heuristic untill the game terminates to determine the utility. Instead, in our limited depth tree a small subtree is created for each new child in order to provide a utility estimate. Because the utility comes from rescues (CT), it is only determined at the bottom level (*rescue solution nodes*), therefore these nodes need to be created. A predefined number of initial children (n_i) are added to each node that are not located at the bottom level. This means that adding a node to the second level (*search solution*) results in $1+n_2+n_2*n_3$ new children in the tree (in case of the four level problem). As opposed to the typical gameplay MCTS simulation process, multiple children are created in our case. This is because multiple HOP samples are required to include the uncertainty of the collaboration problem, and multiple *solution nodes* are necessary to provide initial statistics for the selection process (Section 4.1.1).

4.1.4 Backpropagation

During *backpropagation*, the utility of new children updates the utility and statistics of their ancestors. The utility of the bottom level nodes can be directly computed (as a sum of the utility of each rescue operation according to Equation 3.2) similarly to terminal nodes in the conventional MCTS. Other node utilities in classical MCTS are inferred from their children, most often as the probability of winning the game. This is not different in our case, either the best solution's utility or the average of the alternative problem utilities is chosen, and the utility represents the probability of successfully rescuing all victims. Specifically, the utility of a *problem node* will be the highest utility of its children *solutions* as the children resemble alternative plans, and the planner will choose only the best plan. Likewise, the utility of a *search solution* will be the average of the utilities of its *rescue problem* children, as these children resemble different outcomes of a random process that we cannot control. This average resembles the HOP estimation for the expectation of the utility gain for the UMRTA problem in Equation 3.13. In order to direct the search towards more promising search plans, a *search solution node*'s utility is also partially determined by its own solution quality (the same way as for *rescue solutions*), but with a low weighting factor compared to the utility from the rescue plans.

4.1.5 Winning Action Selection

At the end of the tree growth, the best action is selected. Normally this involves determining the most promising child of the root node. In our case, the result of the process is the collaborative plan of all agents, therefore the action selection needs to propagate through the whole tree.

The *search solution* with the highest utility value is chosen as the search plan in order to maximise the expected utility gain (CD). As for the rescue plan, the best schedule for each HOP simulation needs to be aggregated to find the most beneficial actions the same way as in Chapter 3. The best *search solution* will have several *rescue problem* children of different possible search outcomes (HOP simulations). The best *rescue solution* child is chosen for each of these *rescue problems* that are aggregated according to the method described in the search-aware rescue planning in Section 3.2.2 (Algorithm 1).

4.2 Selection Heuristics

As mentioned in Section 4.1.1, it is not feasible to expand a node with all possible children. As a consequence of this, the selection policy does not only have to make the choice of the child node for traversing through the tree to find the node to expand, but also to decide to expand the current node, or move on to a child. In order to make this decision, an expected reward for expanding a node is calculated. As we are maximising the expected utility by executing the current best plan, the reward corresponds to the expected increase in the overall utility. However, this increase is a result of a costly computation, so it is divided by the required computation time for the expansion:

$$r(n) = \frac{\mathbf{E}\left[\Delta U\right]}{t_{comp}}. \qquad (4.1)$$

Here, $r(n)$ stands for the reward for expanding node n, ΔU marks the utility increase caused by the expansion, and t_{comp} denotes the computation time of the expansion.

The expansion of different nodes – *problem* and *solution* – affects the utility in a different way, because the former explores the decision space, while the latter explores the uncertain process. In the next sections, the expansion of a *problem* (Section 4.2.1) and the expansion of a *solution node* (Section 4.2.2) are detailed.

4.2.1 Problem Node Expansion

When a *problem node* is expanded with additional *solutions* to that problem, the utility of the best solution increases if a better solution is found. Based on our initial tests on different settings from Section 3.3 and other test settings, the utility distribution of the solutions take various forms (e.g. approx. normal, approx. log-normal, multimodal distributions). However, none of the distributions were heavy-tailed, that is not surprising for a random population of values theoretically bounded by the optimal solution. In this case, the right tail – that is the most important for statistics about maximum elements – can be bounded with an exponential distribution. We use an exponential approximation for the tail distribution providing an upper bound (assuming it is not heavy-tailed) to its distribution. Similar upper bound approximation is often used in similar exploration-exploitation problems as MCTS (Kocsis and Szepesvári, 2006).

The expected utility increase in calculated as:

$$\mathbf{E}\left[\Delta U\right] = P\left(\text{within percentile}\right) * P\left(U_{sample} > U_{max}\right) * \mathbf{E}\left(\Delta U \middle| U_{sample} > U_{max}\right). \quad (4.2)$$

Specifically, it is the product of the probability of the random solution to be in the percentile, the probability of the solution being better than the other solutions in the percentile (given it is in the percentile), and the expected distance from the current best solution (given it is in the best solution). The first term is the percentile ratio, ϵ ($P\left(\text{within percentile}\right) = \epsilon$), while the other two terms can be approximated using the exponential approximation.

This exponential approximation is important in the case when the initial SAPT solution has an outstanding solution quality. When the SAPT solution provides better solution compared to the random solutions, the exponential percentile approximation is used to forecast the chance of finding a better random solution than the SAPT schedule. In this case, Equation 4.2 is calculated as follows:

$$\mathbf{E}\left[\Delta U\right] = \varepsilon * e^{-\frac{U_{max}-U_{b,\varepsilon}}{\hat{U}_{\varepsilon}-U_{b,\varepsilon}}} \left(\hat{U}_{\varepsilon} - U_{b,\varepsilon}\right). \quad (4.3)$$

Here, ε, $U_{b,\varepsilon}$, U_{max} and $\hat{U}_{\varepsilon}$ stand for the percentile ratio, the utility at the percentile boundary, the maximum utility and the mean of the percentile respectively.

Otherwise, when the SAPT schedule does not provide the best solution, the probability of finding the best solution can be simply $\frac{1}{N+1}$ (for N existing solutions) regardless of the current best value due to the independent sampling process:

$$\mathbf{E}\left[\Delta U\right] = \varepsilon * \frac{1}{N_{\varepsilon}+1} \left(\hat{U}_{\varepsilon} - U_{b,\varepsilon}\right). \quad (4.4)$$

Here, N_{ε} stands for the number of solutions within the percentile.

4.2.2 Solution Node Expansion

The case of the expansion of a *solution node* is very different. When a higher level (*search*) *solution* node is expanded with *problem nodes*, the utility is not expected to increase as a result. Rather more information is gathered about the utility of the *solution*. This information is useful for not wasting time exploring the suboptimal branches of the tree during the search process.

As described in Section 4.1.1, *problem* and *solution node* expansions need to be compared for the selection process. In order to do this, the information gain needs to be translated into utility increase. First, we present the typical approach during MCTS, a UCB-based heuristic (Section 4.2.2.1). Then, we present our alternative approach based on value of information, that is able to predict the expected overall gain from exploring the uncertainty (Section 4.2.2.2).

4.2.2.1 UCB approach

One approach could be – based on the UCB method (Kocsis and Szepesvári, 2006) – that the expected increase in the lower confidence bound of the node utility gives the value of the information gain. The probability density of the possible node utilities (μ_i) – given the statistics of the children utilities – are marked with different colours in Figure 4.2.

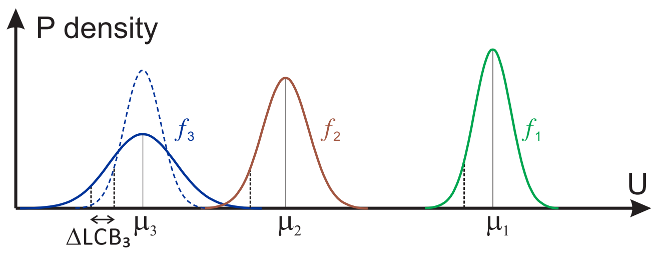


Figure 4.2: Node utility distributions and decision making using lower confidence bound.

The expected increase in the lower confidence bound (LCB) for μ_3 is marked with ΔLCB_3 in Figure 4.2. In this case the original distribution of μ_3 is marked with the solid blue line, while the expected probability density function after expanding node "3" will look like the dashed blue line. The lower confidence bound is marked with a black dashed vertical line for each distributions.

However, Figure 4.2 shows a good example of the weakness of this measure. The highest LCB increase is through expanding "node 3", while the node is clearly suboptimal. This can be observed by the absence of overlap between the density function of "node 1" (f_1) and "node 3" (f_3). This means that the probability of $\mu_3 > \mu_1$ is negligible, so the utility of "node 1" is certainly higher than the utility of "node 3".

It is clear that there is no use in expanding a suboptimal node because the winning action selection (Section 4.1.5) will never choose such a node. The actual information during the expansion of a *solution node* is to identify the best *solution* rather than being certain about the utility of each *solution*, as the LCB approach suggests.

4.2.2.2 Value of Information Approach

In this approach, we try to capture the benefit from the information we get from expanding *search solution nodes*. Such expansion can be beneficial from two perspectives, it can benefit the search planning or the rescue planning process.

In terms of the search plan, the gained information leads to having a more accurate idea of the best *search solution node*. More specifically, if the expansion leads to changing our mind about the best *search solution*, it means that we are not going to waste further effort improving the second best plan. In order to quantify the value of the information gain, we can capture the probability of having a different best node after the expansion. This probability is illustrated in Figure 4.3. After expanding "Node 2", the node's utility distribution (represented with a dark red line) changes. The probability of the new expected utility being higher or lower than another node's expected utility can be calculated before the expansion would take place. The details of this calculation can be found in Appendix A.3. This probability gives information about a possible computational effort for expanding a suboptimal branch. Therefore, this probability can be used to calculate an expected amount of wasted computation time, that can be used as a utility measure.

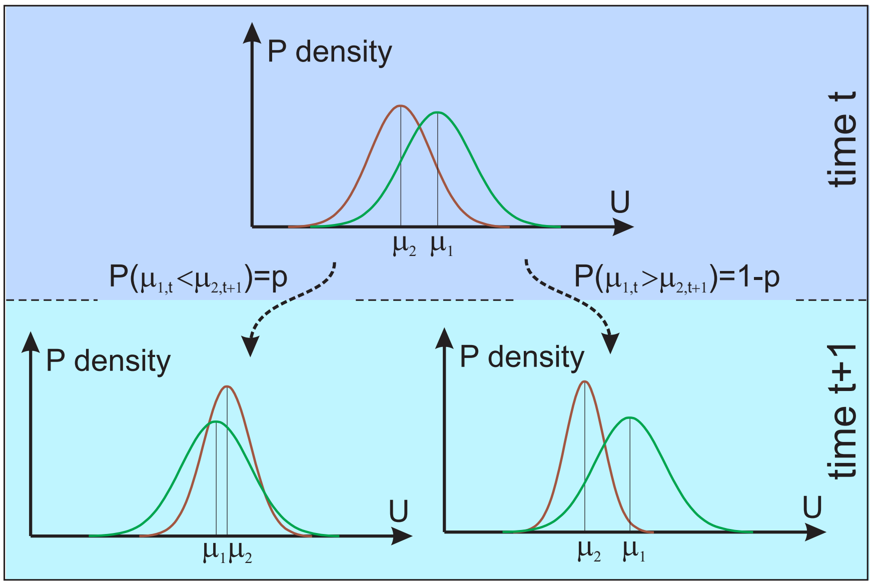


Figure 4.3: Value of information for expanding "Node 2".

In terms of the rescue plan, the gained information leads to having a less noisy HOP plan with a higher number of samples of the random process. By estimating the noise of the acquired motion direction from the best *rescue solutions*, we can calculate the expected decrease in this noise and translate it to a delay in the rescue agents' schedule.

These two components give the value of information from expanding a *search solution*. In the following, we compute the effect of the node expansion on the maximum node utility – that will be the utility of the parent node of these *solution nodes*. The expansions can be divided into two categories according to the conditions for the best node selection.

First Case: Expanding the Current Best Node. First, the value is computed from the search planning perspective. Figure 4.4 illustrates the outcome of expanding "Node 1". The mean utility of the children will change from μ_1 to μ_1'. The probability distribution of μ_1' is represented with a green line, and is according to Equation A.21 in Section A.3.

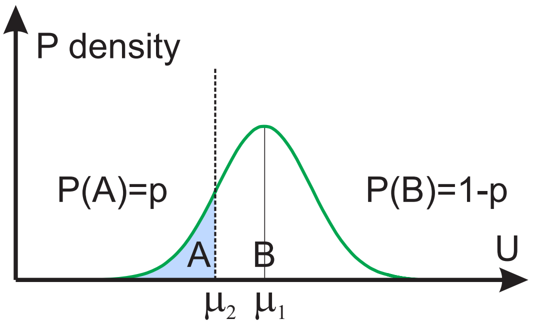


Figure 4.4: Possible outcomes of expanding the current best node ("Node 1")

There are two events defined to classify the possible outcomes of the expansion:

- A: When the new mean will not be the highest any more ($\mu_1' < \mu_2$, where μ_2 is the second best node utility).
- B: When the mean remains the highest ($\mu_1' > \mu_2$).

Event A is the region marked with blue in Figure 4.4.

According to the distribution (Equation A.21), the probability of the above events will be as follows.

$$P(\mathrm{A}) = P(\mu_1' < \mu_2) = \Phi\left(\frac{\mu_2 - \mu_1}{\sigma \frac{k}{n+k}\sqrt{\frac{1}{n} + \frac{1}{k}}}\right) = \Phi\left(-\frac{\Delta\mu}{\sigma'}\right) = p \tag{4.5}$$

$$P(\mathrm{B}) = P(\mu_1' > \mu_2) = 1 - p \tag{4.6}$$

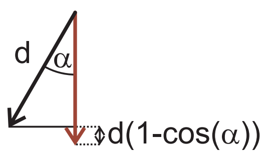


Figure 4.5: HOP direction error effect on rescue agent.

Given this, the expected value of the highest utility value – that is the utility of the parent of these *solution nodes* – after the expansion can be calculated.

$$U'_S = \max(\mu'_1, \mu_2) = \begin{cases} \mu_2 & \text{in case of A} \\ \mu'_1 & \text{in case of B} \end{cases} \tag{4.7}$$

$$\mathbf{E}\left[U'_S\right] = \mu_2 * p + \mathbf{E}\left[\mu'_1 | B\right] * (1-p) \tag{4.8}$$

Using the expectation of a truncated normal distribution (Johnson et al., 1995) the expectation $\mathbf{E}\left[\mu'_1 | B\right]$ can be formulated.

$$\mathbf{E}\left[U'_S\right] = \mu_2 * \Phi\left(-\frac{\Delta\mu}{\sigma'}\right) + \left(\mu_1 + \sigma' \frac{\phi\left(-\frac{\Delta\mu}{\sigma'}\right)}{1 - \Phi\left(-\frac{\Delta\mu}{\sigma'}\right)}\right)\left(1 - \Phi\left(-\frac{\Delta\mu}{\sigma'}\right)\right) \tag{4.9}$$

$$\mathbf{E}\left[U'_S\right] = (\mu_1 - \Delta\mu) * \Phi\left(-\frac{\Delta\mu}{\sigma'}\right) + \mu_1 * \left(1 - \Phi\left(-\frac{\Delta\mu}{\sigma'}\right)\right) + \sigma'\phi\left(-\frac{\Delta\mu}{\sigma'}\right) \tag{4.10}$$

$$\mathbf{E}\left[U'_S\right] = \mu_1 - \Delta\mu * \Phi\left(-\frac{\Delta\mu}{\sigma'}\right) + \sigma'\phi\left(-\frac{\Delta\mu}{\sigma'}\right) \tag{4.11}$$

Where $\Phi(x)$ and $\phi(x)$ are the cumulative distribution function (CDF) and the probability density function (PDF) of the standard normal distribution respectively. Using that $(\Phi(x) - 0.5)$ is odd and $\phi(x)$ is even, Equation 4.11 can be written as:

$$\mathbf{E}\left[U'_S\right] = \mu_1 - \Delta\mu * \left(1 - \Phi\left(\frac{\Delta\mu}{\sigma'}\right)\right) + \sigma'\phi\left(\frac{\Delta\mu}{\sigma'}\right) \tag{4.12}$$

Now, we compute the utility gain estimate in terms of the rescue plan. First of all, we need to understand the drawback from an incorrect chosen direction for a UAV. As can be seen in Figure 4.5, an agent is going to travel in the optimal direction (marked with brown) $d * (1 - \cos\alpha)$ less with α direction error. Assuming that the time increment (Δt, the time period during which replanning is done) is sufficiently low compared to the time necessary to reach nearby tasks, we can define a delay based on this measure:

$$D(\alpha) = \Delta t * (1 - \cos\alpha)\,. \tag{4.13}$$

When the direction error is small due to a sufficient number of samples ($\alpha \approx 0$) this trigonometric function can be approximated by the following polynomial:

$$D(\alpha) \approx \Delta t * \frac{\alpha^2}{2}. \tag{4.14}$$

Now, as we increase the number of samples used in the HOP, the standard error of the average angle (α_m) decreases with the square root of the number of samples (n),

$$\mathbf{E}\left[\alpha_m\right] = \frac{\overline{\alpha}}{\sqrt{n}}. \tag{4.15}$$

This means that this delay decreases linearly with the number of samples (n),

$$\mathbf{E}\left[D(\alpha_m)\right] = \frac{\overline{D(\alpha)}}{n}. \tag{4.16}$$

In order to translate this delay to utility decrease, we need to multiply it by the number of critical tasks in the specific schedule,

$$\Delta U_R(\alpha) = \lambda * D(\alpha) * n_{crit} \tag{4.17}$$

$$\mathbf{E}\left[\Delta U_R(\alpha_m)\right] = \frac{\overline{\Delta U_R(\alpha)}}{n}. \tag{4.18}$$

Finally, when adding k samples to a population of n samples, the following expected rescue utility gain is produced:

$$\mathbf{E}\left[\partial \Delta U_R(\alpha_m)\right] = \overline{\Delta U_R(\alpha)}\left(\frac{1}{n} - \frac{1}{n+k}\right) = \overline{\Delta U_R(\alpha)}\,\frac{k}{n^2+nk}. \tag{4.19}$$

Second Case: Expanding Another Node. Figure 4.6 illustrates the outcome of expanding "Node 2". The same can be applied for this situation as for expanding the maximum nodes. In this case, the mean utility of the children will change from μ_2 to μ_2' after the expansion. The probability distribution of μ_2' is represented with a brown line.

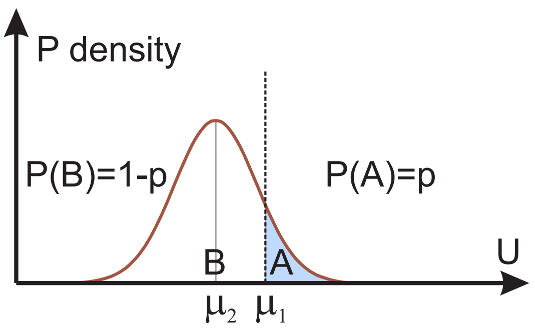


Figure 4.6: Possible outcomes of expanding a lower utility node ("Node 2")

The two events are defined as such:

- A: When the new mean will become the highest ($\mu_2' > \mu_1$, where μ_2 is the second best node utility).
- B: When the other mean (μ_1) remains the highest ($\mu_2' < \mu_1$).

Event A is again marked with blue in Figure 4.6.

The probability of the above events will be as follows.

$$P(\mathrm{A}) = P(\mu_2' > \mu_1) = 1 - \Phi\left(\frac{\mu_1 - \mu_2}{\sigma\frac{k}{n+k}\sqrt{\frac{1}{n}+\frac{1}{k}}}\right) = 1 - \Phi\left(\frac{\Delta\mu}{\sigma'}\right) = p \tag{4.20}$$

$$P(\mathrm{B}) = P(\mu_1' > \mu_2) = 1 - p = \Phi\left(\frac{\Delta\mu}{\sigma'}\right) \tag{4.21}$$

Given this, the expected value of the highest utility value – that is the utility of the parent of these *solution nodes* – after the expansion can be calculated.

$$U_S' = \max(\mu_1, \mu_2') = \begin{cases} \mu_2' & \text{in case of A} \\ \mu_1 & \text{in case of B} \end{cases} \tag{4.22}$$

$$\mathbf{E}\left[U_S'\right] = \mu_1 * (1-p) + \mathbf{E}\left[\mu_2'|A\right] * p \tag{4.23}$$

Similarly to the previous section, the expectation $\mathbf{E}\left[\mu_2'|A\right]$ can be formulated.

$$\mathbf{E}\left[U_S'\right] = \mu_1 * \Phi\left(\frac{\Delta\mu}{\sigma'}\right) + \left(\mu_2 + \sigma'\frac{\phi\left(\frac{\Delta\mu}{\sigma'}\right)}{1-\Phi\left(\frac{\Delta\mu}{\sigma'}\right)}\right)\left(1-\Phi\left(\frac{\Delta\mu}{\sigma'}\right)\right) \tag{4.24}$$

$$\mathbf{E}\left[U_S'\right] = \mu_1 * \Phi\left(\frac{\Delta\mu}{\sigma'}\right) + (\mu_1 - \Delta\mu) * \left(1-\Phi\left(\frac{\Delta\mu}{\sigma'}\right)\right) + \sigma'\phi\left(\frac{\Delta\mu}{\sigma'}\right) \tag{4.25}$$

$$\mathbf{E}\left[U_S'\right] = \mu_1 - \Delta\mu * \left(1-\Phi\left(\frac{\Delta\mu}{\sigma'}\right)\right) + \sigma'\phi\left(\frac{\Delta\mu}{\sigma'}\right) \tag{4.26}$$

In the case when the non-best node is expanded, we expect 0 utility gain in terms of the *rescue solution*, as the plan will be computed according to the best node's *solution*.

Now, Equation 4.26 has the same form as Equation 4.11, with one difference, namely $\Delta\mu$ represents the difference between the current value of the best node and the expanded node, as opposed to the two best node's mean.

Using the resulting formula, the expected search-related utility increase – when expanding the node – can be easily computed:

$$\mathbf{E}\left[\Delta U_S\right] = \sigma'\phi\left(\frac{\Delta\mu}{\sigma'}\right) - \Delta\mu * \left(1-\Phi\left(\frac{\Delta\mu}{\sigma'}\right)\right) \tag{4.27}$$

Note that the value of the current utility is $U = \mu_1$. The reward for a specific expansion can be calculated according to Equation 4.1.

To decrease computation time, the following $f(x)$ function is approximated using a numerical formula for approximating the error function (erf(x)):

$$f(x) = \phi(x) - x\left(1 - \Phi(x)\right) = \frac{e^{-\frac{x^2}{2}}}{\sqrt{2\pi}} - \frac{x}{2}\left(1 - \mathrm{erf}\left(\frac{x}{\sqrt{2}}\right)\right) \tag{4.28}$$

$$\mathbf{E}\left[\Delta U_S\right] = \sigma' * f\left(\frac{\Delta\mu}{\sigma'}\right) \tag{4.29}$$

The utility increase of the HOP is added to this value in case of the best *rescue solution* (Equation 4.19).

4.3 Expansion Heuristics

In this section, we present the chosen random expansion policy for *solution node* expansion is detailed.

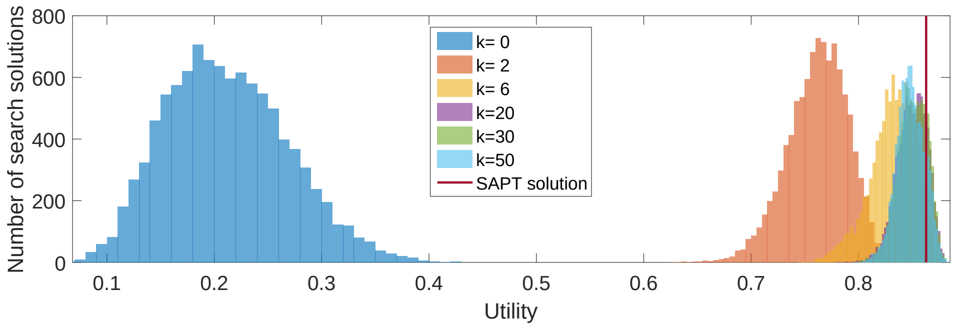
When expanding *problem tree nodes*, random *solutions* are added (as discussed in Section 4.1.2). The *solutions* are constructed in a similar way to the SAPT scheduling (detailed in Section 3.2.1.1). The schedule is grown by assigning one task at a time, but rather than adding the shortest adjusted processing time task, a task is chosen randomly. This results random schedules as required in Section 4.1.2. In order to provide good quality schedules, the random choice is weighted using the adjusted processing time. The weight function (ω) is the following:

$$\omega(d) = \frac{1}{d^k}\frac{1}{C_n}, \tag{4.30}$$

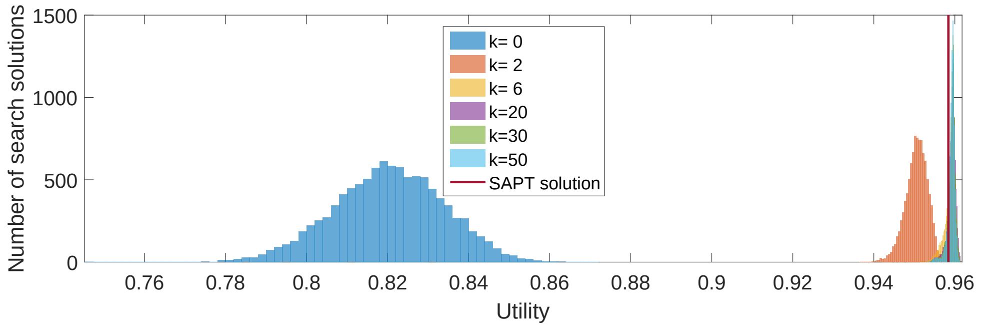
$$C_n = \sum_{\tau \in \mathbf{T}} \frac{1}{d_\tau^k}. \tag{4.31}$$

Here C_n is a normalising factor that assures the sum of weights to be 1. The weight is inverse proportional to the k^{th} power of the adjusted processing time difference of the actual task and the latest scheduled task d. When scheduling for multiple agents, d is the difference from the minimum of the adjusted processing time of the latest scheduled tasks for all agents (the earliest time an agent can be available). We chose this measure for two reasons. First, it is proportional to the adjusted processing time, so if we chose the lowest value each time (the highest probability choice), the result is the SAPT schedule. Secondly, it is always positive and the lower limit is not increasing (as opposed to the adjusted processing time), the value can be close to 0 at any time.

The resulting solutions depend on the value of the k factor. A large value would make the random solutions similar to the SAPT schedule, while smaller values makes the schedules more random (when $k = 0$ is chosen, the schedules are created in a uniform random manner) producing lower efficiency. In order to see the detailed effect of different k values, we have conducted a set of experiments. Figure 4.7 shows the distribution of the solution quality for different values of k. During the experiment, we have generated 10 000 different *search solutions*, each of which has 1000 *rescue problem* children that are solved with SAPT scheduling in order to determine the utility of the *search solutions*. The 10 000 *search solutions* are enough to provide a detailed distribution as can be seen in Figure 4.7, while the 1000 *rescue problem* children provide an accurate utility estimate (10 times more HOP samples as in Section 3.3.1. We can see, that as k increases, the solution quality increases (the utility of the *solutions* become higher), especially the $k = 0$ case that means that tasks are assigned in a uniform random fashion provides low solution quality. After the initial increase, the solution quality distribution converges, there is minor differences between the outcome of different k values.



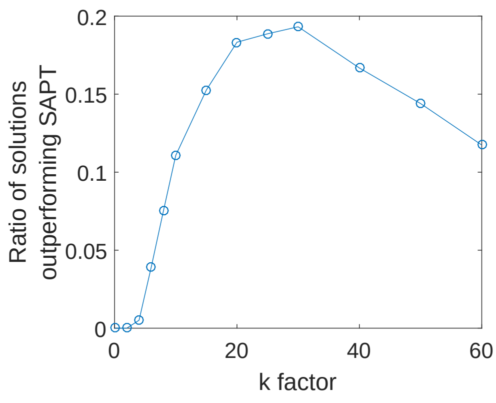
(a) Search solution quality distribution for the initial Ajka planning problem.



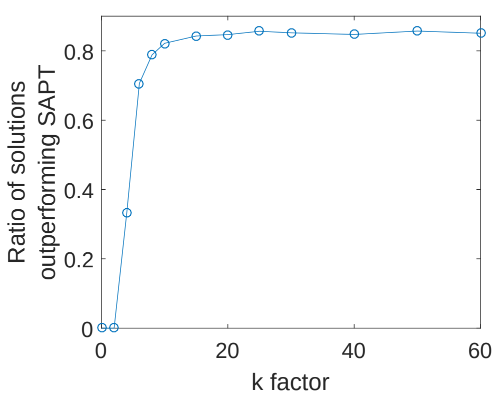
(b) Search solution quality distribution for the initial Haiti planning problem.

Figure 4.7: Comparison of the search solutions with different k values.

In order to compare these cases, we look at the ratio of *solutions* that outperform the SAPT schedule in Figure 4.8. This measure tells what is the chance to provide a better quality *search solution* than the initial SAPT schedule. We can see, that the distribution converges quicker in case of the Haiti problem (around $k = 10$), while in the case of the Ajka problem it tops at higher factors and then the quality starts degrading. This



(a) Experimental results about the search solution quality in the Ajka initial planning problem.



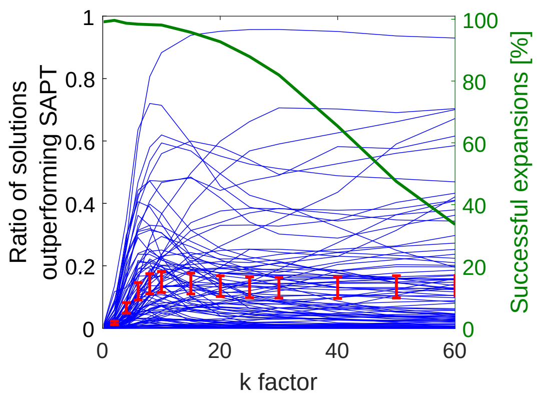
(b) Experimental results about the search solution quality in the Haiti initial planning problem.

Figure 4.8: Comparison of the search solution quality with different k values.

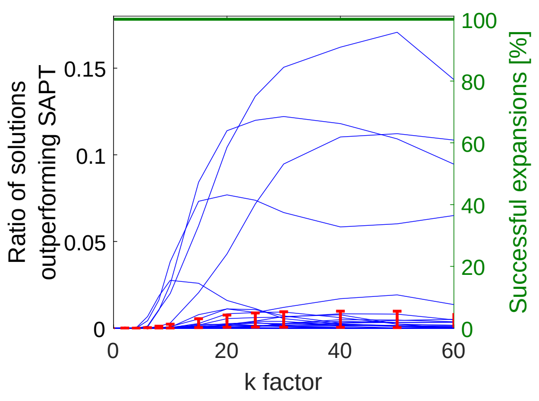
quality decrease is due to the distribution becoming narrower (note the light blue peak in Figure 4.7a) therefore a smaller ratio of solutions provide high utility.

The same comparison is done with *rescue solutions* in a different experiment. We have generated 100 different *rescue problems* for the SAPT *search solution*, and measured the ratio of *solutions* outperforming the SAPT schedule for up to 10 000 different *rescue solutions* for each *rescue problem* for different values of k. The up to 10 000 *rescue solutions* (by up to 100 000 attempts) provide a large enough population to measure the ratio of solutions outperforming the SAPT solution, while the 100 *rescue problems* are chosen for computational reasons, the number of new node creation attempts ($100 *$ 100 000) is the same as in the comparison of the search solutions. The results can be seen in Figure 4.9. Each blue line represents one of the 100 *rescue problems*, while the red marks show the 95% confidence intervals. The green line marks the ratio of successful creation of the 10 000 different *solutions* during 100 000 tries each time. The behaviour is similar to what we observed for the *search solutions*, the ratio of solutions outperforming the SAPT schedule shows an increase as k increases, and then remains on a constant level in both scenarios. Even though in the Ajka scenario there are some *rescue problems* where the ratio peaks at lower k values (around 10) and then the ratio drops, in other cases there is a constant increase resulting the average ratio remaining constant. It is also visible that creating a large number of different *solutions* becomes difficult at higher k values when the number of tasks is lower (20 tasks in average in the Ajka scenario, while 60 in the Haiti scenario).

Next, we further investigated this negative effect of choosing a too high k value. When the k factor is too high, the random solutions become very similar to the SAPT schedule, therefore the chance of finding a unique solution becomes low. In order to test this, we created 50 different random *search solutions* (the average size of an already expanded *search solution* population) and we test the chance of creating a unique random solution.



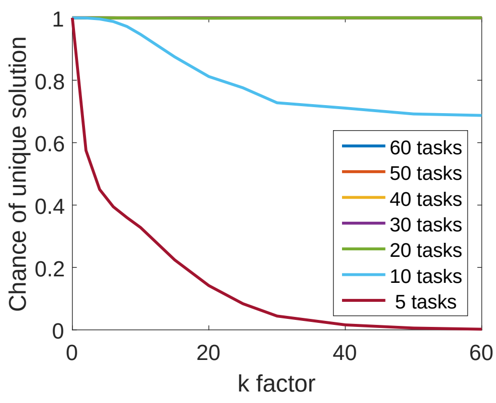
(a) Experimental results about the rescue solution quality in the Ajka initial planning problem.



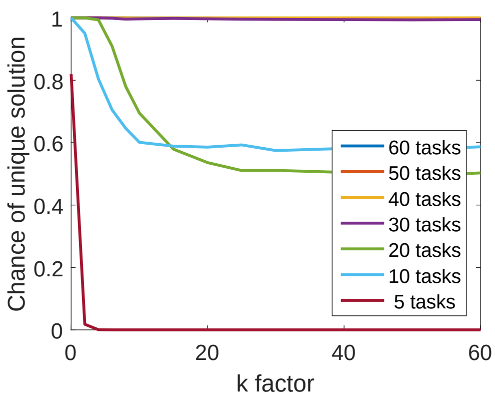
(b) Experimental results about the rescue solution quality in the Haiti initial planning problem.

Figure 4.9: Comparison of the search solution quality with different k values.

Of course this is highly related to the number of tasks, therefore we conducted an experiment with different number of search tasks. This is done using the decomposition planning with HOP approach (Section 3.2.1.2) to advance the search until a specific number of search tasks left. The results of this experiment can be seen in Figure 4.10. As we can see, the chance of finding a unique solution is very high until there are 10-20 search tasks left, but then the number of *solutions* get very limited.



(a) The chance of finding a unique solution to 50 solutions for the partially completed Ajka search problem.



(b) The chance of finding a unique solution to 50 solutions for the partially completed Haiti search problem.

Figure 4.10: Rate of finding new *search solutions* at different k factor values.

Considering these experimental results, we have chosen $k = 20$ that shows a good performance for solving both *search* and *rescue problems*, and the chance of successful expansion is also acceptable. The only disadvantage of this factor is the limited number of unique solutions when the number of tasks are below 10. When the number of tasks are low, the planning problem becomes very simple, and producing a limited number of different solutions easily leads to the optimal solution.

4.4 Empirical Evaluation

The performance of the previously detailed MCTS approach is evaluated in simulations with the same settings as in Section 3.3.1. More specifically, two different versions of the MCTS approach are evaluated in order to compare the UCB and the *value of information* (VoI) approach for the node selection. Both solvers are run with the real-time computation limit (simulation time step in Table 3.3), and a 10-minute initial tree-growth period. These 10 minutes are usually available before flight while the physical platforms are set up. Between simulation steps, the best *search solutions* are preserved to make use of the result of the planning in the previous step, but in order to provide flexibility (RF) these solutions are evaluated in a possibly new problem setting.

More information about the MCTS planning and simulation framework can be found in Appendix B.1.2.

The planning approaches are the following (the approaches from Chapter 3 are also included in the results for comparison):

x1: *Decomposition planning* with gradient descent,

x2: *Decomposition planning* with HOP,

x3: *Unidirectional collaborative planning* along the workflow structure,

x4: *Unidirectional collaborative planning* against the workflow structure,

x5: *Negotiation-based bidirectional collaborative planning*,

x6: *MCTS with UCB*: the *solution node* expansion value is based on the increase of lower confidence bound (Section 4.2.2.1),

x7: *MCTS with VoI*: the *solution node* expansion value is based on the value of information approach detailed in Section 4.2.2.2.

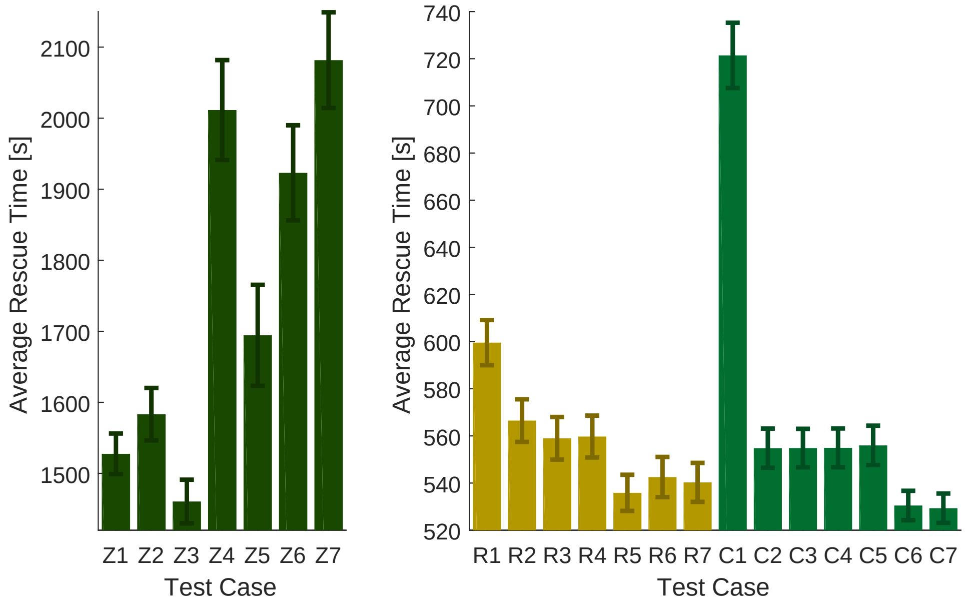
Identically to Section 3.3.2, the notation of the setting depending on available information is the following:

Zx: *Zero information* setting,

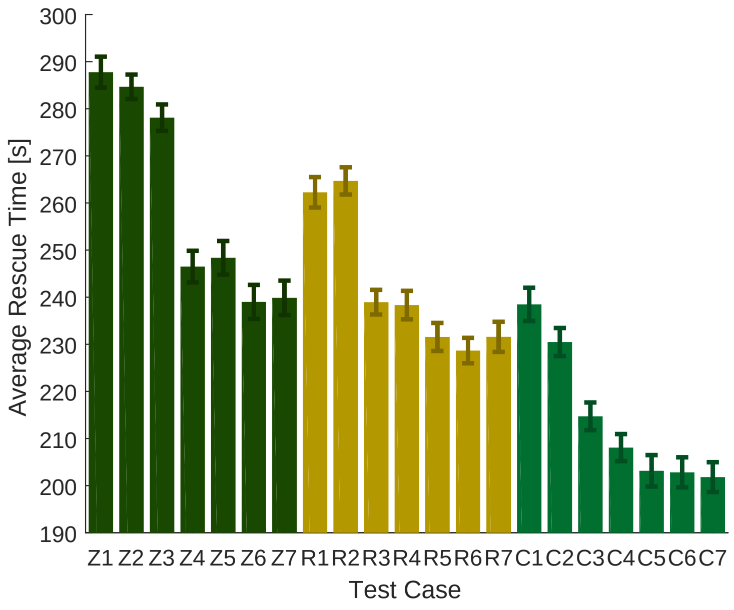
Rx: *Resilient setting*,

Cx: *Complete information* setting.

The evaluation is identical to the previous chapter, with the exception of the method that is the basis of the relative comparison will be the *MCTS with VoI* approach. The measure of the performance remains the same, the average time of the rescue operations in line

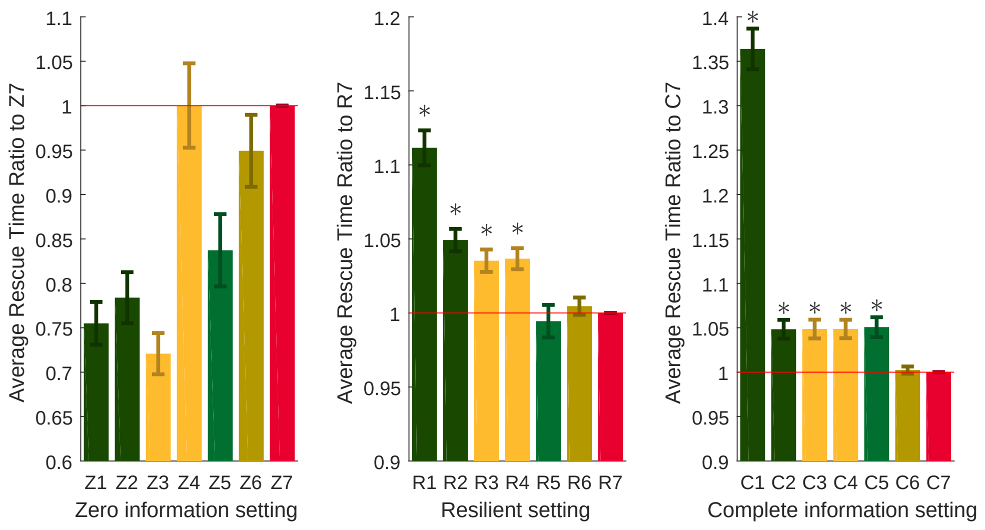


(a) Average execution time of rescue tasks for the Ajka scenario.

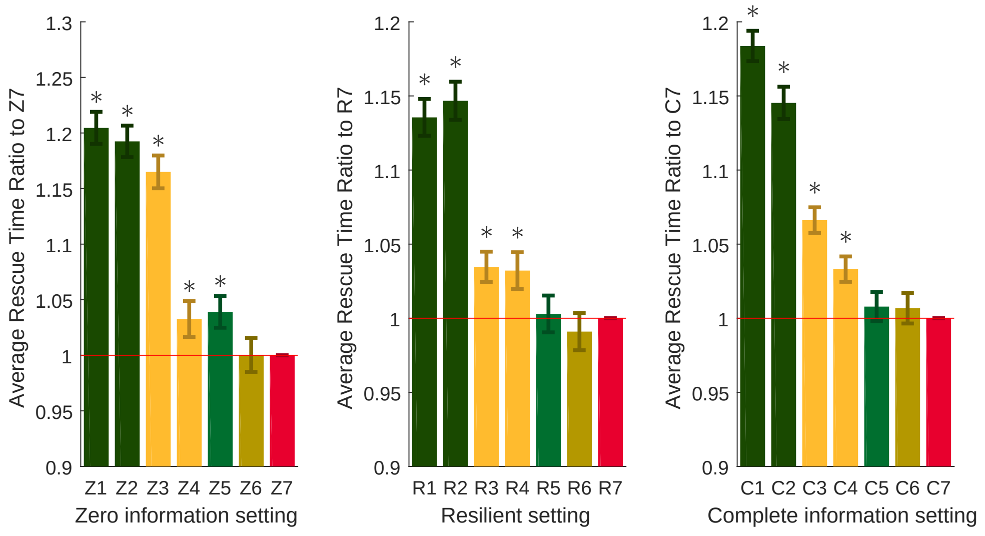


(b) Average execution time of rescue tasks for the Haiti scenario.

Figure 4.11: Overall performance comparison.



(a) Relative performance charts, Ajka scenario



(b) Relative performance charts, Haiti scenario

Figure 4.12: Relative performance compared to the *MCTS with VoI* approach with 95% confidence intervals. Stars mark significant improvement.

with Equation 3.1. The lower this measure, the better the performance. The overall performance comparison can be seen in Figure 4.11, while the relative performance against the *MCTS with VoI* approach can be seen in Figure 4.12. A star above a bar marks significant improvement by the *MCTS with VoI* approach (x7) compared to that approach using a single-sided T-test and a p-value of 0.05.

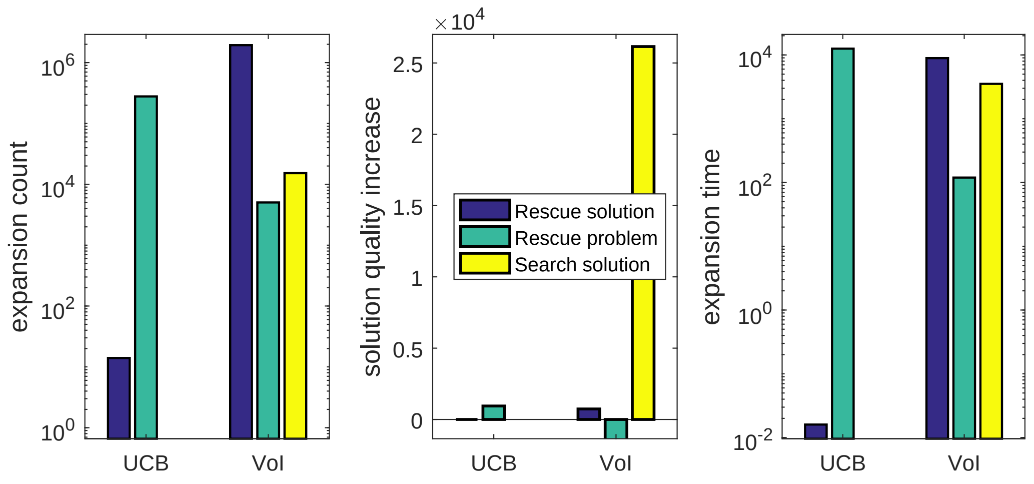
The results show that the MCTS approaches are better able to adapt to situations with low information than the *negotiation-based bidirectional collaborative planner* introduced in Chapter 3. This is due to the dynamic exploration of the uncertainty during the MCTS collaborative planning. When the uncertainty about the rescue is higher, more HOP samples are drawn automatically with MCTS. In more detail, in the Haiti *zero information* setting (Figure 4.12b left), where the *negotiation-based bidirectional collaborative planner* (B5) performs worse than the *unidirectional planning against the workflow* approach (B4), the MCTS planners (B6 and B7) provide the best plan. Unfortunately, the information loss is too large in the Ajka *zero information* case for an advanced planner to be able to optimise over it, the Kullback-Leiber divergence shows this as well, it is the largest value in Table 3.1.

Another advantage of the MCTS approach is that it improves the solution in every aspect. The previous results (Section 3.3.2) show that in the Ajka *complete information* setting, only HOP improves the solution quality, collaborative planning has no effect on the solution quality. In this situation the MCTS approaches (C6 and C7) are able to improve the solution quality by a further 5% (Figure 4.12a right). This is because the planners introduced in Chapter 3 only use the SAPT scheduler for solving the rescue problem, while the MCTS approaches explores alternative solutions as well. Therefore it is able to improve the behaviour of the rescue agents.

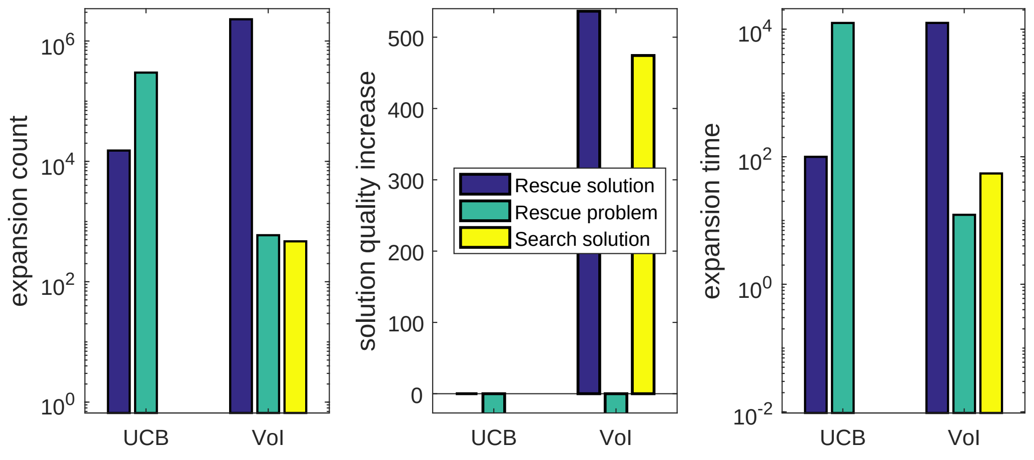
Overall, the MCTS collaborative planning approach represents an enhanced version of the *negotiation-based bidirectional collaborative planning.* Besides providing a scalable anytime approach (RS), the results do not show significantly worse performance in any of the cases (except for the extremely low information Ajka *zero information* setting). Besides this, the MCTS planners perform significantly better in the Ajka *complete information*, the Haiti *zero information* and Haiti *resilient* settings than the *negotiation-based bidirectional collaborative planning* approach due to the above advantages.

However, these results show no significant difference in the performance of the *MCTS with UCB* (x6) and *MCTS with VoI* (x7) approaches. In order to see the difference between these approaches, their behaviour needs to be further investigated.

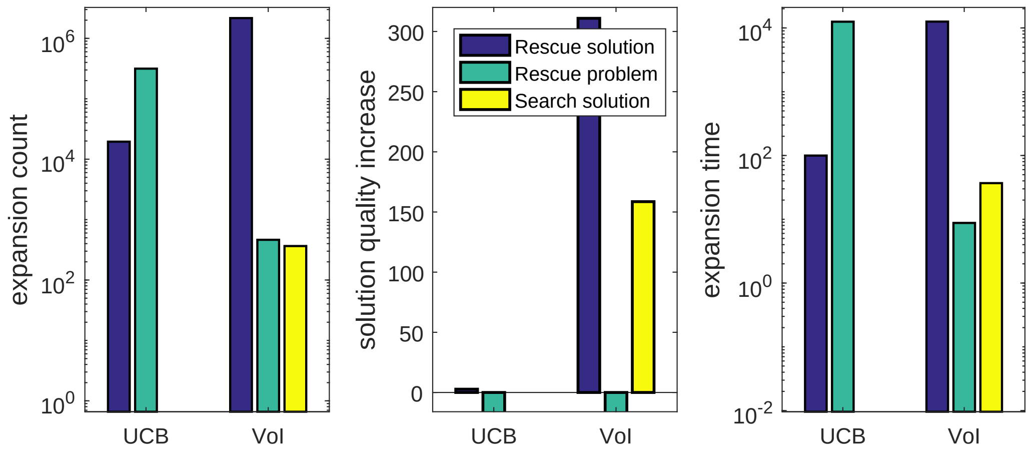
The reason for using the value of information instead of the commonly used UCB is to enhance more valuable expansions to improve the plan quality. In order to understand the difference between the behaviour using the *UCB* and the VoI expansion policy, we need to look at the expansion statistics. To this end, Figures 4.13 and 4.14 show the statistical evaluation of the expansions during an initial tree growth (of 100 s) in all six settings. In detail, the plots in the left column show the number of expansions on the different levels of the search tree (on a logarithmic scale); the middle column plots show the overall solution quality increase as a result of the different expansions; and the right column shows the computation time taken by the different expansions (on a logarithmic scale). The bars with the different colours mark expansions on the different levels of the tree: blue marks *rescue solution* expansions (bottom level); green marks *rescue problem* expansions (middle level); yellow marks the *search solution* expansions (top level). Some negative and large values are outside of the plot range for the solution quality increase plots, so the values can be seen in Table 4.1.



(a) Ajka *Zero information* setting.

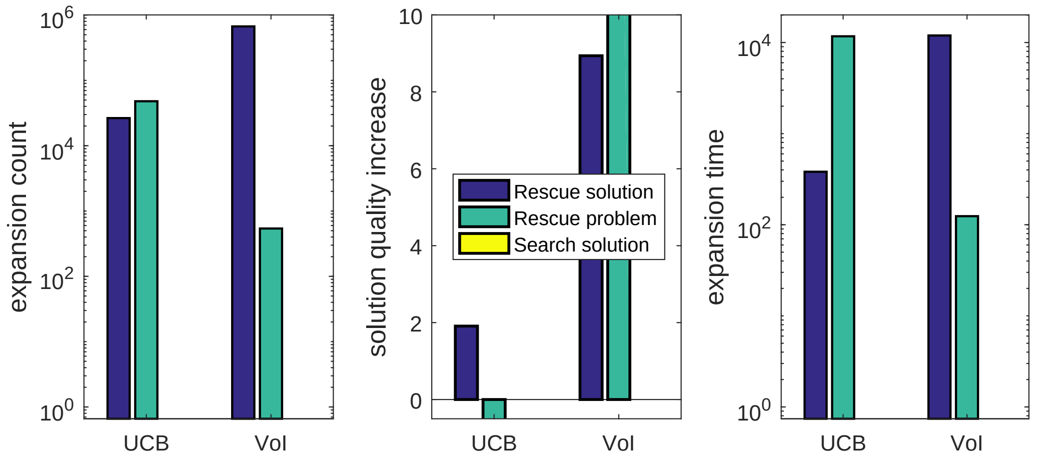


(b) Ajka *Resilient setting.*

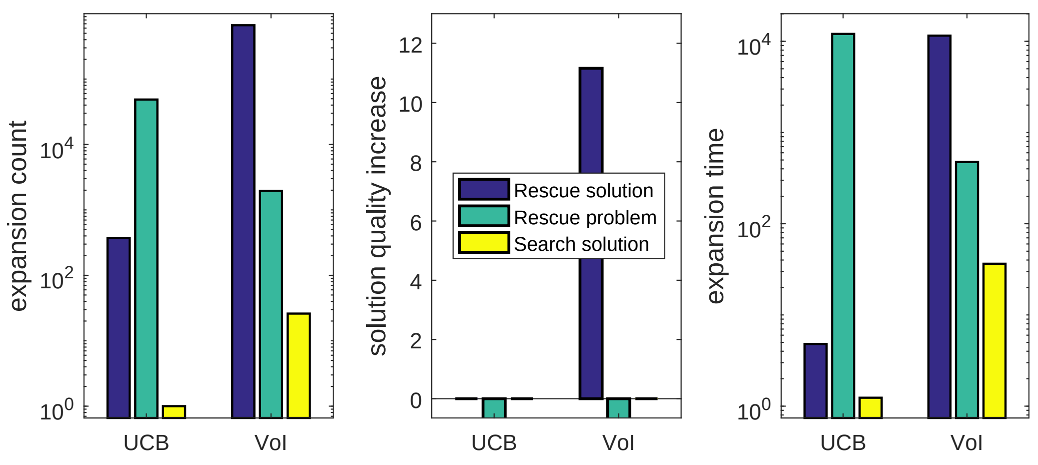


(c) Ajka *Complete information* setting.

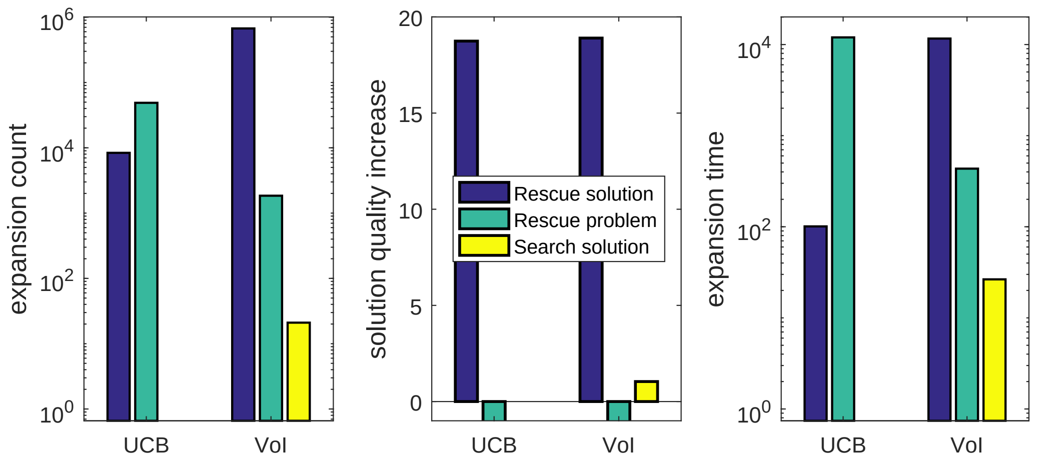
Figure 4.13: Expansion statistics for the Ajka scenario.



(a) Haiti *Zero information* setting.



(b) Haiti *Resilient setting.*



(c) Haiti *Complete information* setting.

Figure 4.14: Expansion statistics for the Haiti scenario.

	Rescue solution		Rescue problem		Search solution	
Information setting	UCB	VoI	UCB	VoI	UCB	VoI
Ajka *Zero information*	0	735.9	942.8	-22 933	—	26 149
Ajka *Resilient setting*	0	536.4	-716.1	-811.8	—	474.3
Ajka *Complete information*	2.748	311.0	-1 011	-623.1	—	158.6
Haiti *Zero information*	18.74	18.90	-344.4	-239.2	—	1.034
Haiti *Resilient setting*	0	11.15	-223.8	-206.2	0	0
Haiti *Complete information*	1.909	8.938	-289.3	42.37	—	—

Table 4.1: Overall solution quality increase from different level MCTS expansions (in seconds over all simulation runs)

First of all, it is clearly visible that the *rescue problem* expansions have sometimes negative effect on the solution performance. This is because these expansions explore the uncertainty, rather than change/improve the joint plan of the agents. It also stands out that expansions on this level are much higher using the UCB approach than using the VoI policy. This is because in situations like the one presented in Figure 4.2, the UCB approach will make suboptimal expansions as it is explained in Section 4.2.2.1 .

As a result of this behavior, the VoI approach will be able to make more expansions on the *rescue solution* and the *search solution* levels, therefore contributing more into improving the joint plan quality. This can be seen in the overall effect of the *search solution* and *rescue solution* expansions in Table 4.1. The accumulated plan quality increase values are 2.7 s (Ajka UCB) against 28 365 s (Ajka VoI) and 20.65 s (Haiti UCB) against 40.03 s (Haiti VoI).

The same tendency can be seen in the computational effort statistics (right column plots in Figures 4.13 and 4.14). The results show that the *UCB* approach spends almost all of its computation time with *rescue problem* expansions, while the *Value of Information* approach concentrates on *solution* expansions in order to improve the plan quality. Unfortunately the resulting improvement in terms of the SAR efficiency is very minor in the presented scenarios, it does not result in a significant improvement by using the VoI approach.

4.5 Summary

In this chapter, MCTS is applied for collaborative planning. The presented approach combines MTCS and HOP in a novel way (Contribution 3 in Section 1.4). Compared to existing work in combining MCTS and HOP, the HOP sampling process is also included in the dynamic tree expansion process. In order to make decisions about these unconventional expansions, novel expectation heuristics are devised based on the value of information approach.

This anytime approach provides a possibility to perform long-term collaborative planning under uncertainty in a flexible way. Besides providing a scalable (RS) anytime approach for the collaborative planning problem, it dynamically provides improvements in the most beneficial parts of the planning problem. As a result, it can cope with a lower amount of information about the uncertainty of the problem, or in some cases provide better quality rescue plans when there is sufficient information about the uncertainty compared to the joint planner presented in Chapter 3. This is also shown through empirical evaluation. Meanwhile, the flexibility for information changes (RF) and the capability to provide a real-time solution (RR) is also achieved by the MCTS planner.

On the other hand, as the MCTS approach only provides significant improvements in some specific settings compared to the joint planner in Chapter 3, the novel heuristics based on the value of information approach do not convey a significant efficiency improvement in the presented evaluation. However, the effects of the novel heuristics are evaluated against the most commonly used heuristic in terms of the quality of the resulting plans. The evaluation shows that the novel value of information heuristic provides significant improvement in terms of the plan quality during the initial tree growth.

The only unfulfilled requirement from the list in Section 1.3 is robust system architecture to cope with unexpected failures (RG). For this reason, the next chapter concentrates on a robust planning system that provides a close to deployment solution.

Chapter 5

Application of Collaborative Search and Rescue Planning

As mentioned in Section 1.1, the mobile phones at a disaster site can be extremely useful to locate, gather information about, and inform victims. In this chapter, we detail an automated victim search system that makes use of the mobile phones of the victims using a combination of existing technologies. By doing this, we explore the following points:

- *How cutting edge GSM technologies combined with UAVs can help collecting vital data for SAR in a matter of hours even at a large-scale disaster settings.*
  The effectiveness of these technologies, are discussed with the main directions for using them in Chapter 1. We provide this example application to evaluate the effectiveness of these technologies in this chapter.
- *How long-term task-based planning is beneficial compared to path planning at a time-critical data collection setting.*
  As mentioned in Chapter 2, the other main methodology for utilising a group of robots besides task allocation is *path planning.* Opposed to the task-based collaboration in the previous chapters, we present a setting where path planning can be compared with a task-allocation based approach. In the evaluation, we show that the previously applied task-allocation based approaches improve the victim search efficiency.
- *How our negotiation-based bidirectional collaborative planning approach (introduced in Section 3.2.3) is applicable in a realistic setting.* We provide a realistic application setting for the SAR collaboration problem (namely decentralised operation, realistic UAV motion model, elaborate observation model). We show that our joint planning approach can be applied in a robust system (RG) and it is able to reproduce the results seen in Chapter 3 under the presented circumstances.

- *How long-term joint planning under uncertainty provides a possibility for further optimisation.* We explore the possibility of using long-term plans, a side product of the planning approaches presented in this thesis, for individual agent behaviour optimisation. During the evaluation, we point out how these plans can improve the overall efficiency, while the same optimisation relying on heuristics is not beneficial.

In the following, we provide a description of the realistic problem setting in Section 5.1. After that, we describe the new observation model of the search agents in Section 5.2. This is followed by the planning approaches used in this setting in Section 5.3. The different planning approaches are then evaluated in a realistic simulation in Section 5.4. Finally our findings are summarised in Section 5.5.

5.1 Problem Setting

The problem setting is slightly different to the one in the previous chapters. The main difference is that we extend the simplistic observation model explained in Section 3.1.3 to better fit to real-life settings. The workflow and the problem structure remains the same SAR collaboration problem (Section 3.1.3), but the search process is different. In particular, search robots instead of taking high altitude aerial photos at certain locations, use a GSM signal detector device to locate mobile phones while traversing though an area. The task of the rescue robots remains largely the same, namely they visit mobile phone locations where they are going to be teleoperated by silver commanders in order to find victims and provide information about their state.

However, the rescue problem is slightly different in this setting. In particular, the information about the mobile phone locations (rescue tasks) is collected gradually rather than instantly as in the previous chapters. It is because the new observation model aggregates constantly incoming information. This means that a dynamic update process is necessary for these rescue tasks. By using an advanced sensor model it is not only possible to provide an estimate for a mobile phone's location, but also for the accuracy of this estimate, therefore the time spent finding the actual location can be estimated. Thus, different process times can be set for rescue tasks. The changes in the rescue planning due to these differences are detailed in Section 5.3.

Similarly to the previous scenarios, we chose fixed-wing UAVs for search. These platforms are be able to carry and power the mobile phone sensing equipment (Munoz-Castaner et al., 2015; Andryeyev et al., 2016). In particular, Andryeyev et al. (2016) concluded that small rotary-wing platforms have a significant drop in the battery life when carrying such equipment. The rescue tasks are very similar to the previous settings, therefore small rotary-wing UAVs remain ideal for those.

The search process in this scenario represents constant measurements rather than visiting predefined locations (as in the previous chapters) therefore the actual path taken by the search robots make a difference. This provides an opportunity for applying *path planning* to direct the search robots as mentioned earlier. In such setting, the exact motion model of the robots becomes more important. In line with this, dynamic motion models were applied during the simulation as opposed to the kinematic models in the previous chapters, making this application closer to a real-life setting in this context as well. In more detail, we applied a limited turn radius, fixed velocity motion model for fixed-wing UAVs as is common in the literature (Scerri et al., 2007; Bernardini et al., 2013; Gan and Sukkarieh, 2011). As for the rotary-wing UAVs, a speed and acceleration limited motion model was applied to accurately capture the capabilities of the safe operation of the platforms in 2D (Erginer and Altug, 2007). Again, this is a widely used model.

5.2 Search Observation Modelling

In this section, we describe our method for observation modeling during the mobile phone scanning process (search). As discussed in Section 2.2.4, the problem of locating RF emitters in Scerri et al. (2007) is very close to the mobile phone scanning problem. For this reason, our signal and detection model relies on the model applied in Scerri et al. (2007) that is based on physical measurements and tested in a field trial. Additionally, we apply a particle filter method (Chung and Furukawa, 2009) in order to maintain the belief distribution, the mobile phone location posteriors, and produce estimates. We found this method the most suitable in the current setting with high variations in the information density. This could be replaced by other probabilistic models (such as Gaussian mixture or grid-based models) in other applications.

To locate mobile phones, we utilise search UAVs equipped with a portable receiver or transceiver, capable of measuring the strength of signals received from a mobile phone, and querying its unique ICCID identifier (Munoz-Castaner et al., 2015; Andryeyev et al., 2016). Given the receiver's location at the time each signal is received, we estimate the location of each mobile phone using a similar approach to Scerri et al. (2007). This process is detailed in the following.

First, for each signal measurement, $m_{x,k}^t$, we estimate the distance $d_{x,k}^t$, between the phone location, $x \in \mathbf{T}_r \cup \mathcal{T}_r^+$, and the location of receiver k, $\mathbf{l}_k^t$, at time t. Due to the high elevation of the receiver, we assume the physical unobscured wave propagation model, the *log distance path loss model* (Rappaport et al., 1996):

$$m_{x,k}^t = m_0 + 10n \log_{10}\left(\frac{d_{x,k}^t}{d_0}\right) + \chi \quad d \geq d_0 \tag{5.1}$$

where m_0 is the mean measurement (in dBm) at an arbitrary reference distance d_0; n is the path loss exponent (which may be empirically estimated); and $\chi \sim \mathcal{N}(0, \sigma^2)$ is zero-mean Gaussian noise with standard deviation σ. By inverting this function, the expected distance for each measurement is thus:

$$\overline{d}_{x,k} = d_0 \cdot 10^{\frac{\overline{m}^t_{x,k} - m_0}{10n}}, \tag{5.2}$$

where $\overline{m}^t_{x,k}$ is equal to Equation 5.1 with $\chi = 0$. Taking all such expectations together, x may be estimated using least-squares multilateration as in Scerri et al. (2007). However, to estimate the optimal path for a search UAV to reduce uncertainty about a mobile phone's location (Section 5.3.2), we require more information about the probability distribution over its location.

With a more advanced probability distribution model, there is an opportunity to extend the work by Scerri et al. (2007) and incorporate information from negative observations as well. In more detail, the mobile phone detector has a sensing range, that means if the signal is below $\hat{m}$ – the sensing threshold – it will result in a negative observation, $m^t_{x,k} = \emptyset$. However, this still provides useful information because it indicates that undetected mobile phones are not close enough to be detected.

Given this, we incorporate both positive and negative observations by defining the *detection likelihood* as follows.

$$P(m^t_{x,k} | \mathbf{l}^t_k, x) = \begin{cases} \phi(m^t_{x,k} | \overline{m}^t_{x,k}, \sigma^2) & \text{if } m^t_{x,k} \geq \hat{m} \\ \Phi(\hat{m} | \overline{m}^t_{x,k}, \sigma^2) & \text{otherwise, } m^t_{x,k} = \emptyset \end{cases}, \tag{5.3}$$

where $\phi(x|\mu, \sigma^2)$ and $\Phi(x|\mu, \sigma^2)$ are the probability density function and the cumulative distribution function of a normal distribution with mean μ and variance σ^2. By Bayes rule, the posterior distribution given all measurements $\mathbf{D} = \{m^t_{x,k} : t > 0, k \in \mathbf{R}_s\}$ for each phone's location is

$$P(x|\mathbf{D}) = zP_0(x) \prod_{t>0} \prod_{k \in \mathbf{R}_s} P(m^t_{x,k} | \mathbf{l}^t_k, x). \tag{5.4}$$

Here $\mathbf{R}_s$ is the set of search UAVs, and z is a normalizing factor to ensure total probability ($\iint_{\mathbb{R}^2} P(x|\mathbf{D}) \mathrm{d}x = 1$).

The posterior for unobserved targets is equivalent, as all observations are negative. This posterior provides the distribution of the uncertain tasks ($\mathcal{T}^+$) in the UMRTA problem of the collaborative victim search problem (Section 3.1.2 and Section 3.1.3). The initial value of the posterior is the belief distribution of undetected mobile phones on the disaster site when the mobile phone is first detected ($P_0(x) \sim Poisson(\lambda(x|\mathbf{D}_x))$). This distribution is periodically updated based on the recent observations, in a similar way to Equation 5.4:

$$\lambda(x|\mathbf{D}) = \lambda_0(x) \prod_{t>0} \prod_{k\in\mathbf{R}_s} P(\emptyset|\mathbf{l}_k^t, x) = \lambda_0(x) \prod_{t>0} \prod_{k\in\mathbf{R}_s} \Phi(\hat{m}|\overline{m}_{x,k}^t, \sigma^2). \tag{5.5}$$

Although the posterior for an observed mobile phone cannot be computed exactly[1], a good approximation can be found using particle filtering. In particular, we adopt a similar method to Chung and Furukawa (2009) except that we do not apply a target movement model, since the victims with distress calls are advised to be stationary. Instead, during the prediction stage, Gaussian noise is added to the particle locations to ensure a dense coverage of the possible locations after resampling.

The resulting observation model is able to provide an accurate posterior distribution of sensed mobile phones for mobile phone location estimates and victim search process time estimates. Besides that, another posterior is calculated about the unobserved mobile phone locations. This information reflects the current state of the uncertain tasks ($\mathcal{T}^+$) in the UMRTA rescue problem (see Section 3.1.2). Additionally, the particle population representing the posterior of mobile phone locations provides an efficient way of computing the effect of taking observations along a specific path (as detailed in Section 5.3.2.2).

5.3 Planning Approaches

In this section, we detail the planning approaches used in the current automated victim search setting. As mentioned in Section 5.1, the search process provides measurements while traversing the disaster site rather than finishing a discrete set of tasks as in the previous chapters. This provides the opportunity to use path-planning to direct the search process. When using path planning, different paths of the robots are compared in order to find the most informative path for each robot. This is most often achieved by tree-search (Section 2.2.3) due to the size of the decision space when looking many steps ahead. Although this technique is able to find specific trajectories that are beneficial during search, it is only capable of planning on a limited horizon. On the other hand, temporal planning or task allocation is able to provide long-term plans over the whole mission time, but requires the reduction of the decision space from custom trajectories to a set of goals. In order to provide these goals, we determine a set of locations that are sufficient to visit in order to cover the area. These locations are located on a grid with a grid size determined by the sensing distance of the GSM sensing equipment in order to ensure coverage and decrease unnecessary redundancy.

As highlighted in Section 5.1, the rescue planning problem is slightly different from the previous settings in two aspects:

[1]The cumulative distribution function of a normal distribution cannot be given in a closed form.

- Firstly, the process times ($P(\tau)$ in Equation 3.12) are not regarded as being uniform, but are estimated by the uncertainty of the mobile phone location. In more detail, the process time estimate is the expected area that needs to be scanned by a spiral search from the mobile phone location estimate divided by the speed of visual search[2]. This can be calculated using the second moment of the distance function from the mobile phone estimate in the posterior probability function (Equation 5.4). After a task has started, the remaining process time is estimated as non-negative and decreasing with time: $P(\tau) = \max(P_0(\tau) - t_{elapsed}, 0)$ where $P_0(\tau)$ is the process time estimate and $t_{elapsed}$ is the time already spent by executing the task.

- Secondly, the dynamic update of rescue task properties needs to be possible. This means that the location and the accuracy of mobile phone locations change as new observations are made about them. The update of these parameters instantly happens if the victim search (rescue task) has not started yet by updating the specific task in the UMRTA problem. When the rescue task is currently in progress the update only happens when it is expected to finish sooner according to the updated estimate. Specifically, the time taken by traveling to the new estimate location plus the new process time estimate is less than the remaining process time ($P(\tau) > P_0(\tau') + trav(\tau, \tau')$). In this case the task will be updated ($\tau \leftarrow \tau'$) and it will be re-initiated by the same robot.

In the following we describe how long-term planning is applied for search in the current context in Section 5.3.1. This is followed by the path planning approaches used to guide the search robots in order to compare their performance with long-term planning approaches in Section 5.3.2.

5.3.1 Long-Term Task-Based Planning

As these approaches are based on the allocation of a set of tasks, the same algorithms can be applied as in the previous settings. An important requirement of the system presented in this chapter is constructing a robust system (RG). In order to achieve this, a certain level of autonomy is granted to the individual UAVs or the groups of UAVs. This means temporary failures in the communication does not cause disturbances on a global level, rather they cause possibly suboptimal behaviour only between the communicating parties. In line with this, we implemented a decentralised version of the collaborative planning algorithms to decrease the impact of message losses especially between different robot groups that typically rely on different communication standards. Unfortunately the decentralisation or segregation of the tree-search based approach (Chapter 4) is very difficult, as the algorithm searches the decision space of all robots at once. This

[2]This is in line with the actual process time detailed in Section 5.4.1.

means keeping the search tree consistent is essential. This can also be seen in the fact that three out of the four iterative steps of MCTS (detailed in Section 2.2.4) include traversing through the tree structure. Decentralising Monte Carlo tree search in the literature is most often done to speed up the calculation process (Chaslot et al., 2008) that is achieved by searching several trees or subtrees at once. Separate tree-search by individual robots would result in inconsistent behaviour when the trees cannot be synchronised.

This leaves us with the approaches presented in Section 3.2 (decomposition, unidirectional and negotiation-based bidirectional collaborative planning). The planning in these approaches is carried out separately for each robot group and this already provides a way for distribution. In order to decentralise the computation between the members of these groups, the typical structure of the groups and the specific algorithms need to be taken into account. The rescue planning process (see Section 3.2.1.2) is composed of two steps, creating HOP plans (using SAPT scheduling) and aggregating them (using Algorithm 1). The decentralisation of this process is achieved by distributing the number of HOP problems solved between the members of the group and then exchanging the relevant information from the solutions between the team members. After this, a partial aggregation is done by each robot to determine its own action. In this way, lost messages result in a lower number of HOP samples processed, that only has a minor effect on the resulting solution quality.

As for the search planning, it is difficult to decentralise the algorithm (Algorithm 2) without causing significant communication overhead. Decentralised approaches exist for the same solution process such as the MURDOCH system (Gasparini et al., 2016) or the repeated auction system by Nanjanath and Gini (2010) but they rely on message passing at each iteration of the algorithm that equals to the number of search tasks. Also there is no significant benefit from decentralising the approach when the number of robots is low (van der Horst and Noble, 2010) and given the special mobile phone sensing equipment carried by these platforms it is not realistic to have a large number of these UAVs. For these reasons, we compute the search planning independently by each search robot and as this is a deterministic process (as opposed to HOP with independent samples) the resulting schedule is going to be identical, resulting in no conflicts between the UAVs.

As discussed in Section 5.1, the motion models of the UAVs are different compared to the simplified ones used in the previous chapters. This can have an effect on the planning process, as the $trav(\tau_a, \tau_b)$ function in Equation 3.12 could be calculated in a different manner. For path planning purposes, the rotary-wing UAVs still calculate path lengths according to the kinematic model, as the acceleration limit has only a minor effect in the actual setting, the UAVs spend most of their flight time traveling at the speed limit. On the other hand, the motion model of the fixed-wing UAVs imposes a significant constraint on the motion of the UAV, therefore we use Dubins curves (Dubins, 1957) for optimal path planning within the dynamic constraints. The length of a resulting path does not only depend on the target location, but also the orientation in which the target is passed by the UAV. This means not only the task schedule, but also the desired heading for each task needs to be determined. In order to cope with this, each search task is split into a number of tasks with different headings (we used 8 for the simulation). The adjusted processing time ($AP(\tau, s)$ in Equation 3.12) is determined for each heading, but for each location only the heading with the shortest adjusted processing time is considered for execution at each scheduling step.

5.3.2 Path Planning

As explained earlier, path planning is able to find informative paths through a disaster area by taking advantage of the maneuverability of the robots. In the following, we detail our benchmark for path planning for search in Section 5.3.2.1. After that, we describe our short-term path planning approach that can be used in combination with long-term task-based planning approaches (detailed in Section 5.3.1) in Section 5.3.2.2.

5.3.2.1 MCTS Path Planning for Search

This approach builds upon the work of Scerri et al. (2007) where multiple fixed-wing UAVs locate RF emitters and decrease the uncertainty of their locations (as explained in Section 2.2.4). This problem formulation is very similar to our search problem, therefore their approach can be applied directly for the path planning of the search robots. In our simulation, we use a MCTS tool (see Section 2.2.4) to reproduce the path planner described by Scerri et al. as it is based on random tree search. Because the task in our case is to find all mobile phones in the disaster area, rather than decreasing the uncertainty of specific RF emitters, we use the intensity of the distribution of unobserved targets (Equation 5.5) as the cost map.

5.3.2.2 Short-Term Path Planning for Search

A further benefit of long-term joint planning is being able to use the other robot's plans for behaviour optimisation. By solving the search with a task-based approach, we restrict the actions of the search UAVs to pass over a grid of locations. On the other hand, a path planner algorithm can take advantage of the maneuverability of the UAVs and find more informative routes for them.

Given this, in this section we show a way to preserve the long-term plans produced by the joint planner, and combine them with a short-term path planner to improve the victim search performance. The empirical evaluation (Section 5.4) shows that doing so improves the overall performance by decreasing the length of rescue tasks, but this is only beneficial when the robots are aware of the plans of other robots (the joint plan).

Initially when using the task allocation based planning, the search UAVs take the shortest path between the scheduled search tasks (locations on a grid). However, sometimes it is beneficial to deviate from this path in order to make more valuable observations about the nearby mobile phones' signals. Of course this means that the rest of the search schedule is going to suffer delays due to the longer path taken to the next search task. In order to assess this benefit and drawback of taking a specific route deviation, the differences have to be translated into utilities.

In order to optimise possible deviations from the shortest path, we use a standard MCTS method (Section 2.2.4) with the information provided by the observation model. This method provides individual optimisation for search agents that can make use of the joint plan of all agents (as mentioned in the beginning of this chapter). In more detail, when taking more informative observations about a mobile phone's location, its location uncertainty (the victim search area, therefore the rescue process time) decreases. In order to approximate this decrease, we use a Monte Carlo method with a three-step process:

1. Possible mobile phone locations are sampled from the current belief from the observation model (Equation 5.4).
2. Mobile phone signals are simulated from these locations along the specific path.
3. The area decrease is estimated by an update step of the particle population based on these simulated signals.

This decrease in the victim search area (rescue task process time) advances the schedule of the assigned rescue robot. It therefore causes this rescue task and the following tasks in the UAV's schedule (the ones not delayed by the search) to finish sooner. If the area decrease is of small increments, we can assume that each of the affected tasks finish the same $\Delta t = \Delta A / v_s$ earlier, where ΔA is the decrease in the victim search area, and v_s is the victim search speed of the UAV. In this case the only information needed from the rescue UAV plans is $\mathbf{E}\left[w(\tau)\right]$, the expected number of consecutive tasks after a specific task that are not delayed by the search process according to Equation 3.14 (as in Section 3.2.2):

$$\Delta U_{\tau}^{+}(\Delta t) = \left(1 + \mathbf{E}\left[w(\tau)\right]\right) * \gamma * \Delta t. \tag{5.6}$$

On the other hand, when making a detour in the route of a search UAV, it is going to cause delays in the further search process. The deviation causes $\Delta t = (s' - s)/v_f$ delay, where s' and s is the length of the deviated path and the original path respectively, and

v_f is the cruising speed of the search UAVs. This delay is the same for all search tasks in the UAV's schedule. The utility decrease comes from increasing the imposed delays on the rescue of the currently unknown mobile phone locations (D in Equation 3.15). If this Δt (the delay from the original plan) increases with small δt increments, the delay increase can be estimated by the number of delayed tasks by search schedule s ($N_{d,s}(\Delta t)$): $\Delta D \approx \delta t * N_{d,s}(\Delta t)$. Because the number of delayed tasks comes from N independent samples of the possible mobile phone locations, division by N is necessary to get the expectation:

$$\Delta U_s^-(\delta t, \Delta t) = -\frac{N_{d,s}(\Delta t)}{N} * \gamma * \delta t. \tag{5.7}$$

Using these two measures, we can quantify the utility difference between two slightly different routes:

$$\Delta U = \sum_{\tau \in \mathbf{T}_r} \Delta U_\tau^+ \left(\frac{\Delta A_\tau}{v_s} \right) + \Delta U_s^- \left(\delta t, \Delta t \right), \tag{5.8}$$

where ΔA_τ is the difference in the victim search area estimate for task τ for the two routes, $\mathbf{T}_r$ is the set of known rescue tasks, and δt ($\ll \Delta t$) is the time difference between completing the two routes. This formula is used as an immediate cumulative reward for a Monte Carlo tree search (MCTS) engine that is run on a time-limited basis (Browne et al., 2012). The tree search uses the standard UCT tree policy that compares the achieved utility gain to the utility gain of taking the shortest path ($c = \sqrt{2} * U_{shortest}$). The outcome of the MCTS is then verified against the original planned path and applied in case it increases the overall utility.

The utility gain calculation is highly dependent on data from the plans of other agents. However, when using a decomposition technique, such data is not available and the robots are not aware of the plans of other robot groups. In these settings, we have substituted the plan-based values with constants and simple heuristics. Specifically, the expected number of delayed tasks is approximated with an average value in the current scenario[3] $\mathbf{E}\left[w(\tau)\right] = w_o$, and the number of delayed tasks is substituted with a simple linear function[4] $N_d(\Delta t) = N_{d0} + \alpha \Delta t$.

5.4 Empirical Evaluation

To evaluate the performance of our automated victim search approach, we chose the 2010 Haiti earthquake (Introduced in Section 1.1). This earthquake is an illustration of

[3]Chosen to be 10 for 12 rescue UAVs and 200 rescue tasks. This means about 17 tasks in average for each UAV, so the average number of tasks following a task is around 8, but we expect already observed tasks to be sooner in a schedule.

[4]The value of N_{d0} does not affect the result, as the delay is compared to the initial (shortest path) solution, α is chosen 7 task/s that is a sensible value based on preliminary simulations.

poor organisation and information distribution between first responders. For example, only half of the SAR sectors could be completed in the first week (MapAction, 2010), also the poor information management made the collaboration of different agencies very difficult (De Ville de Goyet et al., 2011). Given this, we would like to show how our automated victim search approach could have helped save lives by providing crucial information about some of the victims within a couple of hours using a smaller team of first responders.

The search process in the previously detailed system is able to cover a much larger area compared to the Haiti sector search setting in the previous chapters. In more detail, the SAR area contains the whole city of Carrefour (31.45 km^2) as opposed to a SAR sector within the city (0.875 km^2).

In the following, we provide details about the experimental settings in Section 5.4.1, and then discuss the results gained from the evaluation in Section 5.4.2.

5.4.1 Experimental Setup

Our experiments simulate a first response scenario in Carrefour after the earthquake in 2010. In detail, as disaster responders arrive to the site, they send an automated group of UAVs to identify high priority rescue locations for the SAR team. As outlined in Section 5.1, the UAV group consists of fixed-wing search UAVs and rotary-wing rescue UAVs. The fixed wing aircraft carry equipment to create an ad hoc mobile network that allows emergency responders to push messages/calls to the mobile phones in range. The equipment is also able to locate the mobile phones by their signal strength as described in Section 5.2. The emergency responders select the mobile phones that they prioritise given the result of the communication with the individual phones. These selected phones' locations are then automatically visited by one of the rotary-wing UAVs. As a mobile phone location is visited, the victim search process starts around the location. The victim search is carried out by the teleoperation of a silver commander to collect the necessary information for the SAR triage process.

The UN dataset contains the location and the damage status of each building in the area. The damage grading is explained in Table 5.1. We have assigned a chance for each building for a mobile phone to request help in its proximity. There were 15 000+ messages that were made available to the Ushahidi Haiti Project, where the frequency of tags falling in the category of emergency is 5% (Morrow et al., 2011) and Carrefour contains 25% of the tagged buildings in Haiti. These statistics result in about 200 mobile phones requesting emergency help from the 68 596 building locations in Carrefour. The mobile phone locations were determined by a two dimensional random distribution with standard deviation $\sigma = 9.1$ m (the average closest building distance to a building) added to the building location from the dataset.

Grade	Description	P(phone)	#buildings	**E** [#phones]
1	Negligible to slight damage	0.000780	59 377	46.29
2	Moderate damage	-	0	0
3	Substantial to heavy damage	0.004873	2 118	10.32
4	Very heavy damage	0.012182	3 988	48.58
5	Destruction	0.030455	3 113	94.81
Total			68 596	200.0

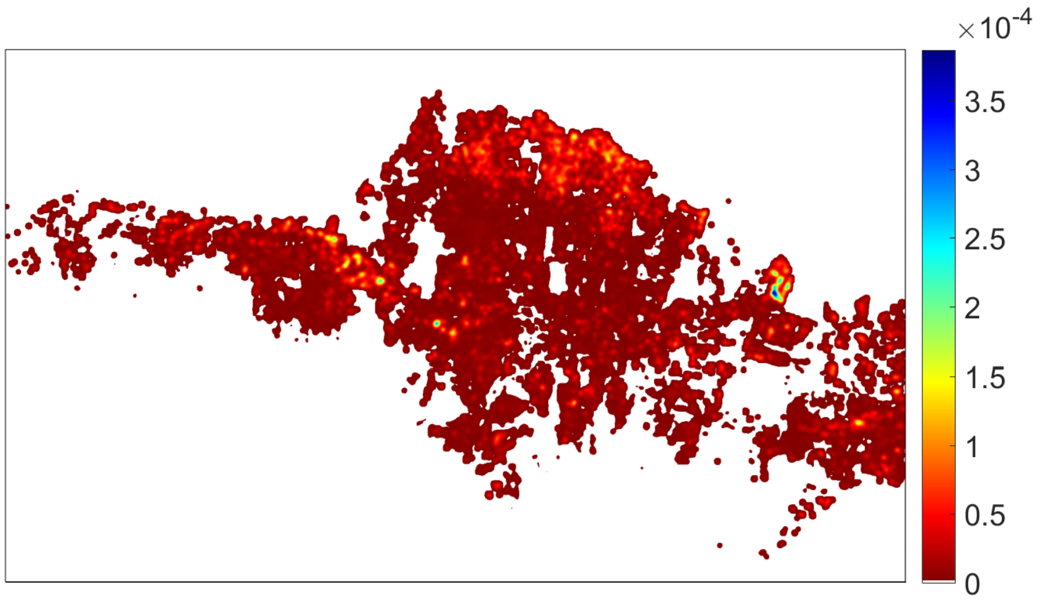
Table 5.1: Haiti assessment (UNOSAT, 2010) damage grades, probability and expected number of mobile phones

Parameter	Fixed-wing UAVs	Rotary-wing UAVs
Maximum speed	22 m/s	10 m/s
Maximum acceleration	-	3 m/s^2
Minimum turn radius	60 m	-
Cruise height	50 m	varying
Mobile phone detector range	300 m	-
Victim search speed (v_s)	-	40 m^2/s
Number of platforms	2	12
Number of tasks	108	avg. 200

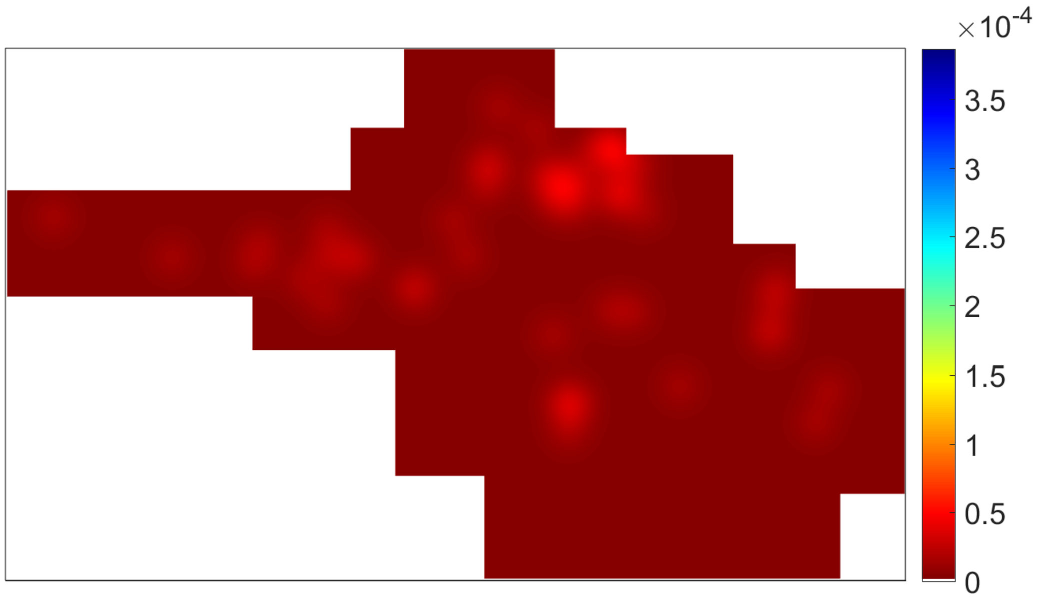
Table 5.2: UAV parameters

The resulting random process to generate the ground truth mobile phone locations for the simulations is an inhomogeneous 2D spatial Poisson point process (Kingman, 1992). The intensity of the spatial Poisson process based on the description above (*ground truth intensity*) can be seen in Figure 5.1a. The robots' initial *belief* of the intensity of the mobile phones at different locations can be seen in Figure 5.1b. Similarly to the Haiti *resilient setting* in Section 3.3.1, this map is generated based on a rough perimeter area of Carrefour and 50 simulated reported locations (e.g. first incoming reports via social media or emergency channels). The locations of these reports are drawn from the *ground truth* Poisson process, and added on the *belief intensity* as 20 times wider Gaussian functions than the ones used for the individual buildings. There is a large difference between the scale of the mobile phone density in the two intensity functions. This is because the high detail in the *ground truth* intensity function includes much higher peaks than the less detailed, smoother *belief intensity* function, although the expected number of mobile phones (the integral of the intensity function) is 200 in both cases.

The performance of the planning approaches is evaluated empirically with 128 different possible disaster outcomes similarly to the evaluations in other chapters (Section 3.3 and Section 4.4). Each outcome is an independent sample from a Poisson process with the *ground truth* intensity (Figure 5.1a), and the approaches are run with the same set of disaster outcomes.



(a) Haiti ground truth intensity [phone/m^2]



(b) Haiti initial belief intensity [phone/m^2]

Figure 5.1: Intensity maps of the Poisson point processes

We have chosen realistic parameters[5] for the fixed-wing and rotary-wing aircraft that can be seen in Table 5.2. In our scenario, once the estimated mobile phone location is automatically reached by rotary-wing UAVs, they are teleoperated by emergency responders to find a victim and collect information for the triage process. The time length of the teleoperated victim search process is estimated during the simulation based on the location error of the mobile phone estimate. We assume the victim search starts at the location estimate of the mobile phone and spirals out from there with constant

[5]The speed of the fixed-wing UAV is adjusted to a specific model with known turn radius compared to the parameters in Table 3.3a

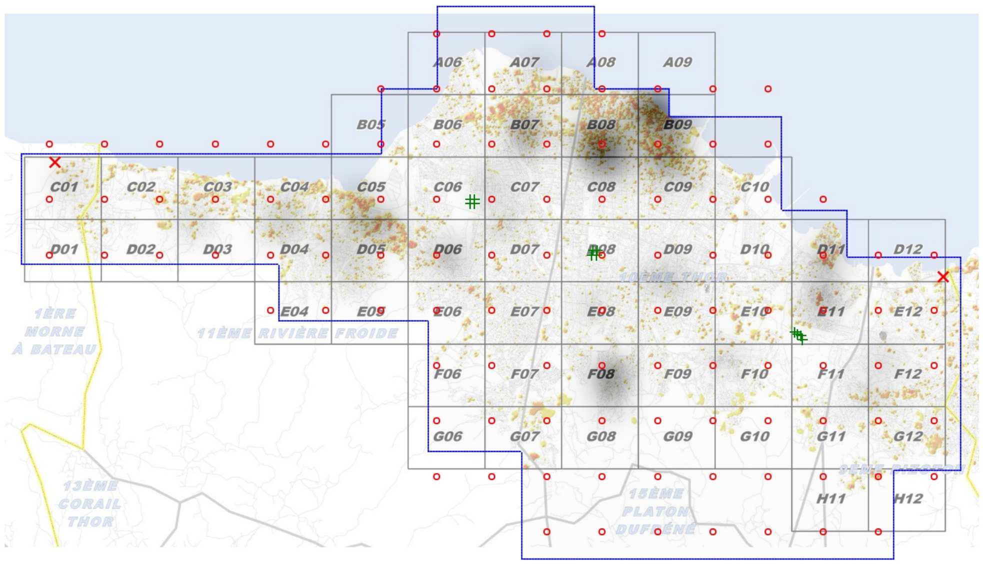


Figure 5.2: UAV start locations and search task locations

speed. Using this approach, finding a victim will take $t_f = e_{loc}^2/v_s$ time, where e_{loc} is the location estimate error distance, and v_s is the victim search speed[6].

We chose a small number of fixed-wing aircraft due to the expensive sensor equipment and the safety pilot requirement for operating these platforms. Moreover, we use a higher number of inexpensive small rotary-wing camera UAVs that are easier to deploy, their number is only limited by the number of possible disaster responders teleoperating them in order to find information about possible victims. The initial mission setup can be seen in Figure 5.2 over the overview of the UN dataset (damaged buildings marked with red and yellow). In detail, the blue edge marks the perimeter of the belief intensity (Figure 5.1b) the dark areas are the high intensity regions in the same map. There are two fixed-wing UAVs that take off on larger roads near the perimeter of the area. These locations are marked with red crosses in Figure 5.2 (in sectors *C01* and *D12*). There are 12 rotary-wing UAVs that take off from 3 locations in manageable-size groups of 4. These locations are larger fields where infrastructure can be set up for controlling them such as a university garden, a college park, and a stadium. The locations are marked with green crosses in Figure 5.2 (in sectors *C06*, *D08*, and *E11*). The predefined tasks for task-based search are marked with red circles and are placed on a grid with 592 m spacing that is twice the detection range (measured on the ground) of a fixed-wing UAV flying at 50 m height[7].

[6]Based on our analysis of a post-disaster UAV footage, we chose the scanning speed equivalent to searching a 20 m wide area with 2 m/s speed.

[7]We ran simulations with different variations of these parameters, but the results were broadly the same.

Name	Search planning	Rescue planning	Joint
MCTS Grad	MCTS based on Scerri et al. (2007)	Gradient descent	No
MCTS HOP	MCTS based on Scerri et al. (2007)	HOP	No
Indep Grad	SAPT MRTA	Gradient descent	No
Indep HOP	SAPT MRTA	HOP	No
Collab	Collaborative MRTA	HOP	Yes
Indep Grad Man	SAPT MRTA & MCTS	Gradient descent	No
Indep HOP Man	SAPT MRTA & MCTS	HOP	No
Collab Man	Collab. MRTA & MCTS	HOP	Yes

Table 5.3: Planning methods used in the compared approaches

The simulation runs separate processes for each UAV and the communication relies on message passing between these processes. The chosen environment[8] is portable (can be run under a range of operating systems) and popular on embedded platforms on robots both in terms of the runtime and the communication. The robots collaborate in a distributed manner as detailed in Section 5.3.1. The states of the UAVs are broadcast periodically and all plans are computed accordingly, therefore the system can rapidly adapt to UAV dropouts. The UAV simulators accept *waypoint commands* that are the standard for open-air UAV coordination. These features result in a system that is very close to physical deployment. More details about the simulation framework can be found in Appendix B.2.

In the following, the results of the conducted experiments are introduced. The statistical significance is tested the same way as in the other chapters (Sections 3.3.2 and 4.4).

5.4.2 Experimental Results

In this section, the performance is evaluated for a range of combinations of the previously detailed approaches in the above settings. Specifically, there are eight different planning methods that show different approaches for search and for how rescue UAVs cope with uncertainty. The planning methods for fixed- and rotary-wing UAVs are compared for the evaluated approaches in Table 5.3.

Our baseline approach, *MCTS Grad*, uses MCTS path planning (Section 5.3.2.1) for search based on the state-of-the-art for RF emitter scanning with multiple fixed-wing UAVs (Scerri et al., 2007); and deterministic MRTA for scheduling rescue using gradient descent when no tasks are assigned to an agent (see Section 3.2.1.1). We evaluate the effects of different planning features in the previously detailed realistic setting:

1. Long-term task allocation planning (SAPT MRTA) for the search (Section 5.3.1 and Section 3.2.1.1),

[8]A Python (`www.python.org`) and NumPy (`www.numpy.org`) based environment using ZeroMQ (`http://zeromq.org/`) for communication.

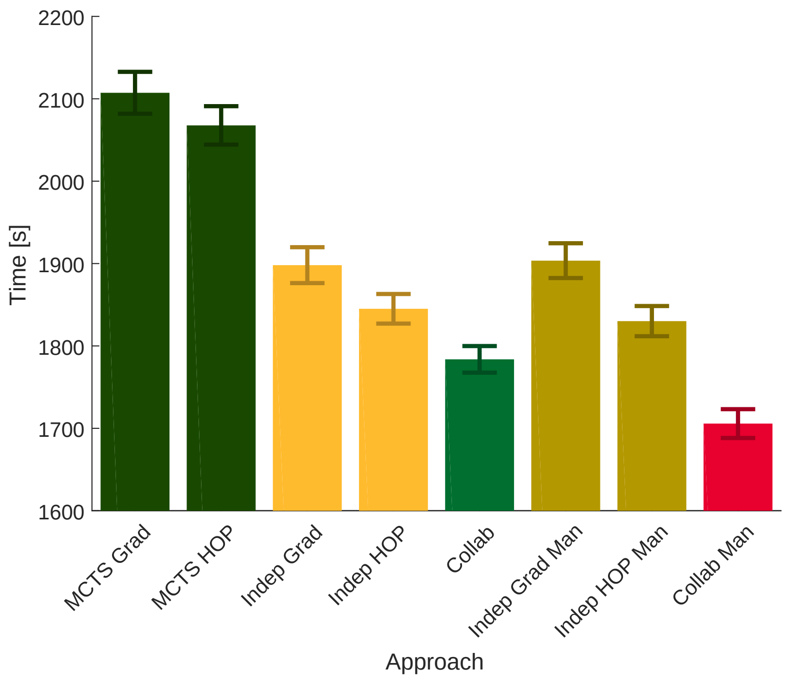
Approach	MCTS Grad	MCTS HOP	Indep Grad	Indep HOP	Collab	Indep Grad Man	Indep HOP Man
MCTS Grad		-	-	-	-	-	-
MCTS HOP	**0.02**		-	-	-	-	-
Indep Grad	≈ **0**	≈ **0**		-	-	0.15	-
Indep HOP	≈ **0**	≈ **0**	**0.001**		-	**0.001**	-
Collab	≈ **0**	≈ **0**	≈ **0**	≈ **0**		≈ **0**	≈ **0**
Indep Grad Man	≈ **0**	≈ **0**	0.29	-	-		-
Indep HOP Man	≈ **0**	≈ **0**	**0.0008**	0.079	-	**0.0003**	
Collab Man	≈ **0**	≈ **0**	≈ **0**	≈ **0**	≈ **0**	≈ **0**	≈ **0**

Table 5.4: Result difference p-values (significance in bold)

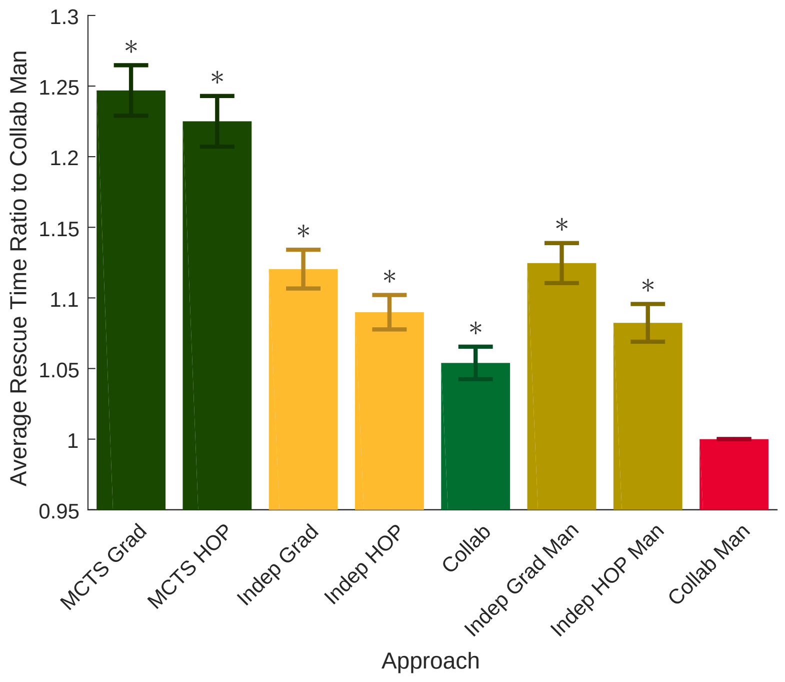
2. Hindsight optimisation (HOP) for planning under the uncertainty of mobile phone locations with rescue UAVs (Section 3.2.1.2),
3. Collaborative (joint) planning of the search and the rescue using a negotiation-based approach (Section 3.2.3),
4. Long-term planning with short-term path optimisation using Monte Carlo tree search (MRTA & MCTS) for search (Section 5.3.2.2).

The overall performance comparison can be seen in Figure 5.3, while the significance of the comparison of approaches can be seen in Table 5.4. All the presented planning features improved the overall performance compared to the baseline approach:

1. The long-term planning allowed the search to explore the area more efficiently, not leaving isolated areas behind. This improved the overall performance by 12% (*Indep Grad* vs *MCTS Grad*).
2. Hindsight optimisation significantly improved the performance in the independent planning settings compared to the gradient descent approach (*Indep Grad* vs *Indep HOP*: 3.3% and *Indep Grad Man* vs *Indep HOP Man*: 5.1%). This is because long-term planning under uncertainty provides a possibility for rescue UAVs to split up and cover different areas rather than all travel towards the first mobile phone locations. This is especially important when covering a larger area (the same can be observed in the Ajka setting in Section 3.3.2). HOP is also a necessary step for collaborative planning with the UAV teams, because the connection between the search and the rescue problem lies within the not yet searched regions.
3. Joint planning of the long-term plans also increases the performance. There is a 6.9% improvement without the fixed-wing path optimisation (*Indep Grad* vs



(a) Average rescue time



(b) Relative performance comparison with *Collab Man* (significant difference marked with star)

Figure 5.3: Overall performance comparison with 95% confidence intervals.

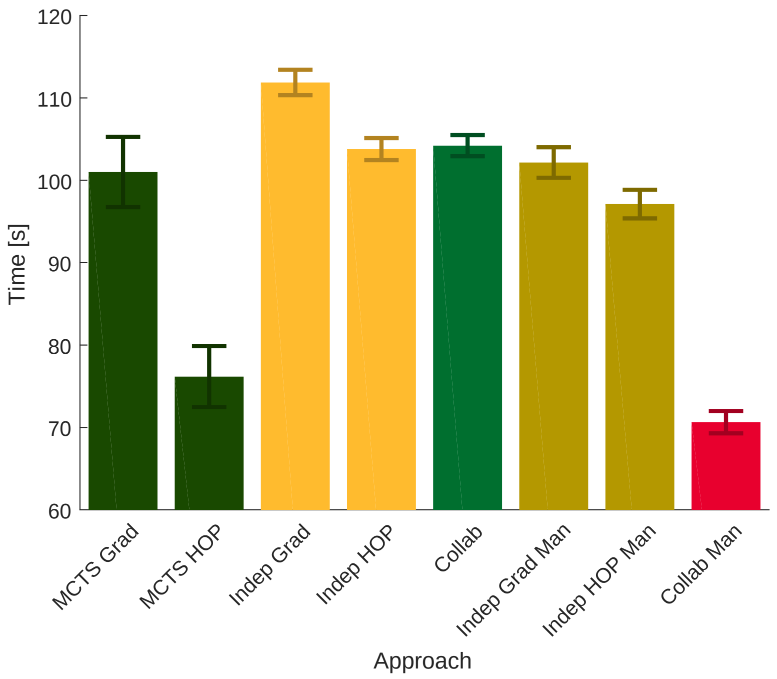


Figure 5.4: Average time a rotary-wing UAV takes to find a victim from arriving to the estimate mobile phone location (rescue task process time).

Collab) and 12.5% with the optimisation enabled (*Indep Grad Man* vs *Collab Man*). This is in line with the results in Chapter 3 (Section 3.3.2).

4. The short-term fixed-wing path optimisation significantly improved (by 5.4%) the performance when it could take advantage of the plans of the teams of the UAVs to prioritise important search or rescue tasks (*Collab* vs *Collab Man*). Also, there is no significant improvement when the plans are substituted with heuristics (*Indep Grad* vs *Indep Grad Man* and *Indep HOP* vs *Indep HOP Man*). This shows how being aware of the full plan can be beneficial when using individual optimisation techniques. This information can identify the priority of actions in relation to the actions of others, therefore makes such optimisation techniques more efficient.

Altogether, there was an 24.7% improvement between the baseline approach and the approach with all enhancements. The simulations were done while keeping the real-time computational limit using a single core of an Intel Xeon E5-2670 processor per UAV. The long-term planning was done every 10 s, while the short term planning every 1 s.

Providing some insight into the behaviour of the approaches, Figure 5.4 shows the average process time of rescue tasks, the average time rotary-wing UAVs took to find victims near the sensed location of mobile phones. There is a clear improvement in this measure when the fixed-wing path optimisation is added, but only when it can make use of the joint plans of the UAVs (*Collab Man*). This shows the importance of knowing the long-term plans of the agents rather than relying on general heuristics when applying further optimisation in their behaviour. It also explains the lack of improvement

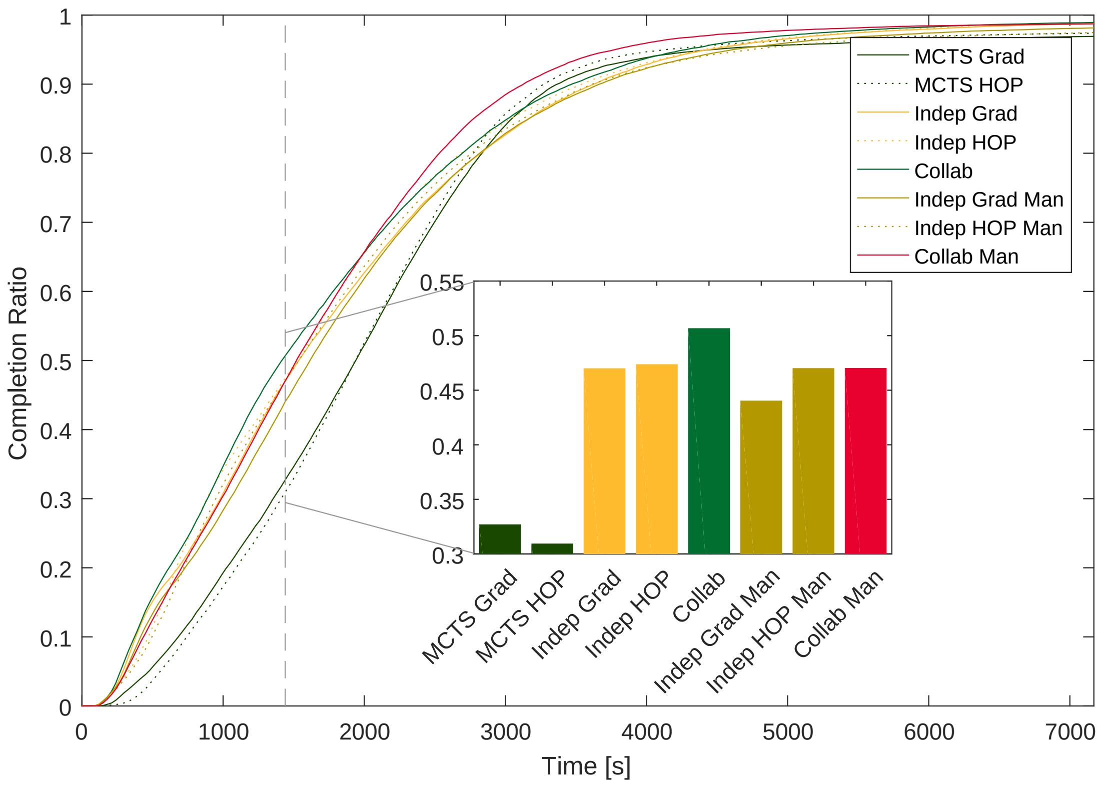


Figure 5.5: Completion rate of finding victims over time. Bars show the progress after 24 minutes.

when the path optimisation is used in other approaches. The difference in the rescue task length for the baseline and the tree-search with hindsight optimisation approaches (*MCTS Grad* vs *MCTS HOP*) emphasizes the main difference between the gradient descent and HOP behaviour. Specifically, when using gradient descent, rescue tasks are immediately scheduled as mobile phones are observed, while during HOP rescue UAVs wait at high mobile phone density locations until they are searched. Having waited, there are more observations made about other mobile phones that decreases the time the rotary-wings need to find a victim at these locations, but waiting imposes a large delay causing a low overall performance.

The progress of the automated victim search can be observed over time in Figure 5.5. The completion ratio shows the ratio between found and not yet found victims. The bar graph shows a cross section after 23 minutes, the maximal flight time of a popular professional small rotary-wing camera drone (DJI Phantom 3 Professional, Table 1.2 in Section 1.2). If the victim search would stop after this time, the collaborative approach without path optimisation (*Collab*) would find the most victims, 101 on average. The fact that only half of the victims are found at this point shows that the long battery life of the platforms is crucial for a successful application of automated victim search in a similar setting. This can be solved with using larger UAVs with internal combustion engines, but both the transportation and pricing of such platforms prevent the deployment of a larger group. Also these larger platforms require safety pilots, as they can cause serious injuries or death during a malfunction.

In more detail, the progress using tree search path planning based search (*MCTS Grad* and *MCTS HOP*) shows a significantly less steep completion curve in the first 20 minutes due to the slower coverage of the area by search. It can also be seen that when using deterministic planning for the rotary-wing planning (*MCTS Grad*, *Indep Grad*, *Indep Grad Man*) the immediately scheduled tasks accelerate the initial progress in the first 500 seconds, but the progress rapidly falls back as this causes the UAVs to move to the same regions that is suboptimal for covering all areas. The progress graph also shows the differences between the joint planning with and without the fixed-wing path optimisation (*Collab* and *Collab Man*). In particular, the path optimisation will slightly delay the search process in order to make critical mobile phone locations more accurate. In accordance with this, the progress has an approximate 2-minute delay at t=1000 s. However, after that the completion stays steep for a longer time due to the more accurate mobile phone locations; the completion rate is 3.7% higher (in average 7 more victims found) at t=3000 s.

5.5 Summary

In this chapter, we presented an application of collaborative SAR planning in disaster response, in an automated victim search setting. The automated victim search consists of scanning an area in order to locate mobile phones and then finding victims near the mobile phone locations with two specialised groups of robots (UAVs) for each tasks. We have applied the collaborative SAR problem in this setting where search agents locate mobile phones, and rescue agents find victims near these mobile phones.

To this end, we have explored the points stated in the beginning of this chapter:

- In the presented large scale automated victim search setting, an area more than 30 km^2 was scanned and 94 % of the victims (188 in average) were identified within one hour. This performance is outstanding compared to the current approaches of rapid assessment using UAVs (OCHA, 2014). The occupied resources also suggest that this can be achieved with a moderate sized first responder team. This was achieved by using two fixed-wing UAVs with special GSM equipment (Andryeyev et al., 2016), and twelve small rotary-wing UAVs. The necessary crew consists of ten to twelve silver commanders, two safety pilots and some technical staff for setting up the UAVs and infrastructure.
- We have shown that in a multi-UAV target search scenario long-term task-based planning outperforms path planning approaches, especially when combined with a task allocation problem as in the collaborative SAR problem. Path planning approaches can be very beneficial when specific routes needs to be found in an

environment with obstacles, because the feasibility of the route is highly dependent on minor features of the path. On the other hand, when path planning is applied for making observations, where the utility gain is not driven by minor features of the path, path planning approaches are better in providing coverage therefore perform better overall.

- The evaluation has shown that the bidirectional negotiation-based approach can be used and improves the efficiency in the presented highly realistic setting (decentralised operation, realistic UAV motion model, elaborate observation model).
- The evaluation has also shown that long-term joint planning for a complex problem provides the benefit of providing accurate data about the other agents' intentions. We have successfully used this data for enhancements of the search, by individual short-term path planning with the search agents preserving their long-term plans. The evaluation shows that such optimisation is not beneficial when the data is not available but substituted by heuristics. On the other hand, when it can take advantage of the plans of other agents, it provides a significant improvement.

We have evaluated the performance of the planner in a simulation system that is close to deployment. We have applied an accurate sensor model of mobile phone signal strength, simulated the control of the UAVs resulting in realistic flight paths, and also simulated the synchronisation and communication between the UAVs with improved robustness (RG) through decentralisation. The resulting system provides a portable, easy to deploy solution for embedded hardware (flexible runtime and environment for communication) on physical robots (waypoint UAV commands). During the evaluation, our optimised joint planning approach finds victims 25% faster than the state-of-the-art path planning approach. The collaboration approach relies on the *negotiation-based bidirectional collaborative planning* (Section 3.2.3) that is flexible for incoming information (RF), and provides real-time planning (RR). This has been applied in a robust decentralised manner (RG) in line with Contribution 4 in Section 1.4.

Chapter 6

Conclusions and Future Work

This thesis develops a number of new approaches for heterogeneous autonomous robot collaboration in complex settings such as search and rescue. We now summarise the contributions of this work and give directions for future work.

6.1 Conclusions

Challenging planning problems are often simplified with relaxed constraints to apply complex algorithms to optimise system performance. Such simplifications are often necessary to avoid an optimal or near-optimal solution to be intractable. These simplifications may include replacing direct dependencies of actions with general goals. When multiple actions create a workflow process, this would mean optimising each stage independently in order to optimise the whole process. However, there are problems where the dependencies between the individual actions are very strong, therefore a specific sequence is required to reach a goal. These problems can be dealt with temporal planning, that searches for beneficial action sequences given a general problem description, although it is often difficult to provide a problem description in a realistic setting. This is especially true in a multi-robot setting, where there is significant uncertainty present. The sensors of the robots constantly update the robot's belief of their surroundings (e.g. new tasks are discovered, new estimations are gathered about existing tasks or about the environment). This means that the current state of the problem dynamically updates, and the future problem states can be estimated using a probabilistic model. In this work, we explore multi-robot settings with a workflow process between the actions of different robots in a setting where these dependencies are often relaxed. We show that these action dependencies can be taken into account using collaborative planning under uncertainty.

This approach is shown in a heterogeneous robot collaboration problem in a disaster response setting. We identify four *criteria* for efficient planning in this context (as discussed in Section 1.3). In particular, the heterogeneous robots take *different roles* in a workflow process (CH), the uncertain information about the disaster site is available as a *probabilistic model* (CP), the setting is *time-critical* as the chance of a successful rescue decreases over time (CT), and an *uncertain planning* approach needs to find the best actions according to the probabilistic model (CD).

Besides these criteria, we also identify key *requirements* for an applicable system in real-life conditions. Such systems are able to provide very useful tools for disaster responders and contribute into saving lives after a disaster. These requirements are *flexibility* for incoming data and dynamic changes in the information (RF), the ability to run in a *real-time* manner on a portable infrastructure (RR), *scalability* in order to cope with disasters that can be represented with higher problem complexity (RS), and *robustness* against failures in the system (RG).

In Chapter 2, we assess the related literature on collaborative multi-robot planning against these requirements and determine if any of the approaches are able to tackle problems outlined by the given criteria. In so doing, we find that existing approaches tackling similar multi-role collaborations usually decompose the collaboration problem into simpler problems that are solved independently. In some cases a sequential (i.e. unidirectional) approach is applied to create a collaborative plan in settings where the collaboration plays a significant role. These approaches are extended with our novel joint (i.e. bidirectional) planning approach that improves the partial plans of the collaboration problem to create a joint plan.

To evaluate this approach, we constructed a problem with real-life applications in disaster response, the SAR collaboration problem (Section 3.1.3). Collaborative planning in this problem is challenging, because the coupling between two planning problems lies within the uncertainty of the problem. In order to provide plans under uncertainty (CD), we apply a probabilistic determinisation method, HOP (Contribution 1 in Section 1.4). This way we are able to produce long-term plans under uncertainty, that is essential for collaborative planning. Using this method, we construct an online planner that is the first to provide joint plans for a multi-role collaboration of robots under uncertainty (Section 3.2.3, Contribution 2 in Section 1.4). We evaluate this approach against different planning approaches based on simulations using data from two different disasters. The evaluation shows that in most settings collaborative planning outperforms the current state of the art planning approaches. The presented approach provides high flexibility to incoming information (RF), and is computationally feasible in the presented settings (RR). However, we show that the planner does not provide a scalable solution (RS), the complexity of this approach is a third degree polynomial of the number of search tasks that will exceed the likely computational budget for larger problems (>200-300 search tasks).

In order to increase the scalability (RS) of the joint planning, we propose an anytime algorithm in Chapter 4. The algorithm is based on a probabilistic tree-search algorithm, MCTS combined with HOP to provide joint plans under uncertainty. We provide an extension to the existing work on combining MCTS with HOP (Contribution 3 in Section 1.4). Our novel approach is able to explore the uncertainty in the same dynamic manner as the decision space is explored by conventional MCTS. We are able to compare the plan improvement from exploring the decision space and the uncertainty of the problem using a value of information approach (Section 4.2.2.2). Using the same evaluation as in Chapter 3, we find that the MCTS approach is able to further optimise the SAR efficiency and adapts better to situations with low information.

Aiming at constructing a system that can be applied in real-life conditions, an application of collaborative planning in automated victim search is explained in Chapter 5 (Contribution 4 in Section 1.4). We create a system where cutting-edge sensors and UAVs are simulated in order to carry out rapid victim localisation and information collection. Compared to the previous evaluation scenario (Section 3.3), a larger area (an entire city) is searched, providing an efficient tool for information collection for first response (as detailed in Section 5.4). Although the system is still evaluated in simulation, the simulation provides sufficient details to be able to efficiently direct physical UAV platforms. This includes physical sensor modeling, dynamic simulation of UAV platforms, and implementation in a close to deployment runtime. The more realistic sensing provides an opportunity to compare collaborative long-term planning against path-planning approaches that are often used for search. There is also considerable effort spent on preparing the system for faults (RG) including the decentralisation of the planning to be able to cope with lost messages. We evaluate our collaborative planning approach against other task-based and path-planning based approaches and show that our approach finds victims 25% faster compared to the state of the art. Moreover, when using long-term joint planning, the plans of other robots can be used to compute the effect of certain actions in a global level. This allows optimisation techniques to perform better. Specifically, we show that a short-term path planning technique is only beneficial when it is aware of the plans of other robots as opposed to when heuristics are applied to provide missing information.

When taken together, this thesis defines an important problem for using multi-robot systems for aiding disaster response. We provide a novel approach to this problem that resembles a joint online planner for the uncertain multi-role robot collaboration problem. Our initial approach (Section 3.2.3) provides a flexible (RF) and real-time (RR) approach. We then provide an extension using MCTS to provide a scalable solution (RS). Finally, we show the approach's potential in a realistic simulation of a large-scale automated victim search setting. The presented solution provides a system that is robust (RG) and easy to deploy.

Although the algorithms in this thesis solve the SAR collaboration problem defined in Section 3.1.3, the extension to other problems (fulfilling the identified criteria) is relatively simple. We chose this specific problem because it provides a good basis for heterogeneous multi-UAV collaboration and has significant applications in disaster response.

One possible alternative domain would be wildlife preservation. As outlined in Mulero-Pázmány et al. (2014), the fight against poachers has a great potential for multi-UAV systems. The time-criticality (CT) of this setting is also a strong factor in order to prevent successful poaching activities and catch poachers. The workflow process of uncertain actions can be considered in several scenarios. First, a collaborative scenario can be considered between fixed-wing and rotary-wing UAVs. Fixed-wing platforms are applied for long-term patrolling or identification of the movement of endangered species (possibly using a radio tracker providing similar setting as in Chapter 5) while rotary-wing UAVs are used for rapid response for identifying poachers (or possibly as a deterrent tool as pointed out in Mulero-Pázmány et al. (2014)). Another scenario would include the collaboration of UAVs and rangers in confronting poachers. In this setting, UAVs would be used to identify poaching activity, while rangers would have to approach poachers before they succeed. Both of these settings provide applications for the SAR collaboration problem, and collaborative planning could prevent successful poaching thus save animals from poachers. On the other hand, the game-theoretic implications of this setting cannot be ignored. For this reason, as soon as the poachers' motion is considered, the problem becomes significantly more complex.

Another domain for similar time-critical setting with great potential of using UAVs is power line inspection (Katrasnik et al., 2010). A setting of the time-critical identification and recovery of a fault provides a good use-case for a collaborative system. A rapid mapping of the power line in order to detect the cause of the fault can be carried out by one agent group and the actual recovery can be achieved using another agent group. This structure also suits the SAR collaboration problem that if solved using collaborative planning will reduce the reaction time to the incident thus contributes into fixing the failure earlier. This results in a higher system reliability by reducing the system downtime.

Besides solving the SAR collaboration problem, there are other implications of the contributions of this work. First of all, our novel approach for applying HOP for uncertain task allocation (Contribution 1 in Section 1.4) can be applied in various obstacle-free multi-robot task allocation settings. For example, UAVs in open air conditions usually assumed to travel in an obstacle-free environment, and in realistic settings often only probabilistic model is available about the upcoming tasks.

Moreover, our approach to combine HOP with long-term planning (Contribution 2 in Section 1.4) provides a useful tool for uncertain planning when dependencies lie within the uncertainty of the problem. The SAR collaboration problem is a good example for this, but there are other problems with the same characteristics. Firstly, *collaboration of autonomous underwater vehicles* (AUVs) is difficult due to limited communication abilities under the water surface (Belbachir et al., 2010). There can be a setting where rendezvous points need to be planned besides a sequence of actions with uncertain time length (e.g. due to unknown underwater drag conditions or uncertain task complexities). HOP with long-term planning can be very useful in such setting to come up with plan policies to provide minimum delays and successful information exchanges during rendezvous between multiple AUVs. Secondly, *probabilistic multi-pursuer planning* in pursuit-evasion games also shows similarities (Hespanha et al., 1999). In such settings, multiple pursuers collaborate to maximise the chance for capturing one or more evaders. The motion of the evader can be modeled as an uncertain process that provides a connection between the utility of plans of the pursuers. If the evader is already captured in an earlier point of the plan by another pursuer, there is no benefit for visiting a location that the evader would have visited later. This also provides an appropriate use for the combination of HOP and collaborative planning in order to improve the capture efficiency.

Moreover, our novel combination of MCTS with HOP (Contribution 3 in Section 1.4) provides a possibility to improve MCTS in its original domain of gameplay, in incomplete information games[1]. Determinisation using HOP has been criticised in multiplayer incomplete information games, as there is no effort made for collecting more information or considering information known by opponents (Cowling et al., 2012). However, in solitaire games this approach can tackle uncertainty successfully. For example, the work of Bjarnason et al. (2009) can be extended by dynamic decision making about HOP expansions using a value of information heuristics similar to the ones explained in Section 4.2.2.2.

Finally, we demonstrate the applicability of collaborative planning in a highly realistic setting in disaster response (Contribution 4 in Section 1.4). Our intention with this contribution is to provide a way how advanced sensors, cutting-edge UAV technologies and advanced planning are able to make an impact in disaster response. The contribution provides a vision for combining available technical resources to make an automated victim search system that is able to provide key information to first responders within a couple of hours after arriving to the disaster site.

6.2 Future Work

As this work provides a system for collaborative planning with UAVs in a disaster response domain, there is much more to be done in order to construct a system that can

[1]HOP cannot be applied in games with complete information, as there is no uncertainty from the perspective of a player.

be used to aid disaster response. Specifically, the integration of the described technologies in physical UAV platforms is necessary to identify key technical challenges for a successful deployment in disaster response settings. Another important issue is to cooperate with disaster responders in order to integrate the technologies with their practices. Finally, we discuss the opportunities for creating a general framework for collaborative planning.

6.2.1 Field Trials on UAV Platforms

The system described in Chapter 5 tries to bring the collaborative planning concept close to deployment is several ways. Firstly, it uses independent processes for each UAV with message passing between them to establish the collaborative planning. Secondly, the output of the planning process is a set of waypoint commands, the same format that physical UAV platforms accept. These commands are executed by simulators that follow the behaviour of an autopilot expected on the physical platform. Thirdly, observation modeling includes the physical model of the GSM sensing equipment with independent signal generation and processing. Finally, the implementation is done in a portable runtime that is available for embedded devices that can typically be carried by UAV platforms.

However, there are several aspects that are very difficult to provide without a reliable system in place with physical UAV platforms. These include safety aspects of communication faults, exact (possibly multi-channel) communication layout including communication between different UAV platforms, adjustment of the sensor model based on experimental data, and anomalies in the behaviour of system components. Trials of the system using physical platforms also show the robustness of a system in circumstances that are unlikely to be encountered in simulation.

6.2.2 Integration Into Disaster Response Systems

When targeting a system that can be used by disaster responders, the current practices of emergency response teams is a disaster situation cannot be ignored. As described in Section 1.2, disaster responders are involved in the collaborative SAR system, therefore their impact on the performance needs to be taken into account. As concluded by Goodrich et al. (2008), data representation and user interface plays an important role for the efficient use of the introduced technologies. According to this, the interface and the possibly small details about the exact system behaviour can make a large impact on the overall efficiency. For example, the planning process might sometimes fail to consistently choose the best plan from several plans of similar quality (e.g. sweep search in different directions or swapping plans of two agents). It could make a significant difference how this is communicated towards a supervisor to avoid confusing them. Another important issue is to gain the trust of the operators in the system. This can be affected by many factors (Chen et al., 2011) but consistent and reproducible system behaviour of the system is most likely an advantage in this context. Altogether, there are several aspects where small changes in the planning process or the provided information about plans could make a significant impact when trialing the system with emergency responders.

In order to successfully integrate a multi-UAV system into their work, research needs to work closely with emergency responders while developing and testing systems with increasing complexity. Large-scale emergency response trainings such as Angel Thunder (Jones, 2014a) provide a great opportunity to trial and evaluate a new tool for emergency responders in real-life situations without risking actual lives.

6.2.3 Generalisation of Collaborative Planning Under Uncertainty

As discussed in Section 6.1, there are other problems where collaborative planning under uncertainty can be successfully applied. In order to fully explore similar domains, a framework for describing and evaluating such problems needs to be created. A generic description language similar to the ones used in temporal planning (Fox and Long, 2011) can be created to explain an uncertain collaborative planning problem. The set of different actions and their dependencies can be described the same way, but it needs to be extended with the description of the uncertainty in the problem. While the rest of the problem description is constant during a scenario, the information about the uncertainty dynamically changes, as it becomes certain once an action is completed. As discussed in Section 1.3, the uncertain information can be successfully described using a probabilistic model. This probabilistic model provides the distribution of a random variable. The possible outcomes of a random variable can vary between different applications (e.g. a set of tasks in the SAR collaboration problem, the time necessary to travel between two locations with an AUV or a path an evader takes over time). According to this, a general description is necessary to define different random variables, and a mechanism is needed to update the probabilistic model based on the observations. In a resulting framework, different determinisation and planning approaches can be compared in different problem domains.

Appendix A

Supplementary Proofs and Algorithms

A.1 Decision making to determine the optimal motion direction of rescue robots given hindsight plans

In this section, the HOP solution aggregation method is detailed that is applied in Algorithm 1 in Section 3.2.1.2. The problem is directing rescue robots while mobile phones (and therefore rescue tasks) are not explored yet – the search process is ongoing – in order to maximise the overall utility by finding victims sooner.

In this case, the optimisation should rely on the distribution of the possible mobile phone locations. As detailed in Section 3.2.1.2, the distribution is sampled and several solutions are given using HOP. Consequently, the aggregation problem is to find a movement direction for each robot given the hindsight schedules for each sample of the tasks in order to maximise the expected utility gain of the rescue agents.

At first, we solve problem for single task planning horizon, then we extend the solution to multiple tasks.

Theorem A.1. *The optimal aggregated direction for an agent to maximise the utility of executing the closest task from a random distribution is the average of the directions of closest tasks from independent samples of the task distribution.*

Proof. The proof is shown for an example of two tasks. In Figure A.1, we can see an agent choosing a direction for proactive movement to execute one of the two possible tasks appearing with p_1 and p_2 independent probability respectively. The distance from the tasks are d_1 and d_2 respectively, $d_1 < d_2$. The unit vector to task 1, task 2 and the chosen direction of proactive movement is $\mathbf{v_1}, \mathbf{v_2}$ and $\mathbf{v_a}$ respectively. We assume a linearly decreasing utility function by the distance from the task, $U = U_0 - d * \lambda$ (in line with Equation 3.1). The optimal $\mathbf{v_a}$ vector can be derived as follows:

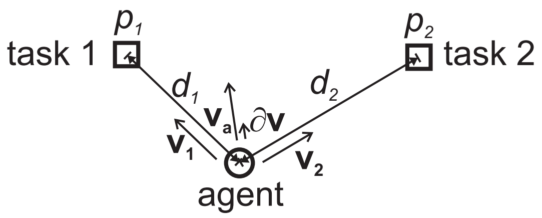


Figure A.1: Simple proactive movement example.

$$E[U] = p_1 * (U_0 - d_1 * \lambda) + p_2 * (1 - p_1) * (U_0 - d_2 * \lambda), \tag{A.1}$$

$$\frac{\partial E[U]}{\partial \mathbf{v}} = \lambda * \left(-p_1 * \frac{\partial d_1}{\partial \mathbf{v}} - p_2 * (1 - p_1) * \frac{\partial d_2}{\partial \mathbf{v}} \right), \tag{A.2}$$

$$\frac{\partial E[U]}{\partial \mathbf{v}} * \frac{1}{\lambda} = -p_1 * \frac{\partial |d_1 \mathbf{v_1} - \partial \mathbf{v}|}{\partial \mathbf{v}} - p_2 * (1 - p_1) * \frac{\partial |d_2 \mathbf{v_2} - \partial \mathbf{v}|}{\partial \mathbf{v}}. \tag{A.3}$$

Now, as the length of $\partial\mathbf{v}$ converges to zero, we can make the following approximation:

$$\partial |\mathbf{v_n} - \partial \mathbf{v}| \approx - \frac{\mathbf{v_n}}{|\mathbf{v_n}|} \cdot \partial \mathbf{v}, \tag{A.4}$$

that is the length of the parallel component of $\partial\mathbf{v}$ with $\mathbf{v_n}$. Therefore,

$$\frac{\partial E[U]}{\partial \mathbf{v}} * \frac{1}{\lambda} = p_1 * \frac{\mathbf{v_1} \cdot \partial \mathbf{v}}{\partial \mathbf{v}} + p_2 * (1 - p_1) * \frac{\mathbf{v_2} \cdot \partial \mathbf{v}}{\partial \mathbf{v}}, \tag{A.5}$$

$$\frac{\partial E[U]}{\partial \mathbf{v}} * \frac{1}{\lambda} = \frac{(p_1 \mathbf{v_1} + p_2 * (1 - p_1) \mathbf{v_2}) \cdot \partial \mathbf{v}}{\partial \mathbf{v}}, \tag{A.6}$$

$$|\partial \mathbf{v}| = \partial d, \quad \partial \mathbf{v} = \mathbf{v_a} \partial d, \tag{A.7}$$

$$\frac{\partial E[U]}{\mathbf{v_a} \partial d} * \frac{1}{\lambda} = \frac{(p_1 \mathbf{v_1} + p_2 * (1 - p_1) \mathbf{v_2}) \cdot \mathbf{v_a} \partial d}{\mathbf{v_a} \partial d}, \tag{A.8}$$

$$\frac{\partial E[U]}{\partial d} = \lambda (p_1 \mathbf{v_1} + p_2 * (1 - p_1) \mathbf{v_2}) \cdot \mathbf{v_a}. \tag{A.9}$$

Now, to maximize the expected utility gain, we have to choose a direction of movement $\mathbf{v_a}$ that makes $\frac{\partial E[U]}{\partial d}$ the highest, that will point to the same direction as $p_1\mathbf{v_1} + p_2 * (1 - p_1)\mathbf{v_2}$, as it will maximize the scalar product in Equation A.9:

$$\arg\max_{\mathbf{v_a}} \frac{\partial E[U]}{\partial d} = \frac{p_1 \mathbf{v_1} + p_2 * (1 - p_1) \mathbf{v_2}}{|p_1 \mathbf{v_1} + p_2 * (1 - p_1) \mathbf{v_2}|}. \tag{A.10}$$

If we take infinite number of samples of the possible outcomes, task 1 will be chosen with a ration of p_1, and task 2 will be chosen with a ratio of $p_2 * (1 - p_1)$, and no tasks will be in the $(1 - p_1) * (1 - p_2)$ ratio of the samples. The average direction vector in this example will be identical to the optimal direction in Equation A.10. $\square$

The average of the vectors similarly lead to the optimal proactive movement direction for more tasks or agents when only one task can be performed by an agent. The proof consists of the same steps for more tasks or more agents, and we leave this for the reader.

Theorem A.2. *The optimal aggregated direction for an agent to maximise the utility of a sequence of tasks is the weighted average of initial directions from hindsight optimisation solutions for independent samples of the task distribution. The weights are the number of non-delayed tasks in the corresponding schedule.*

Proof. The technique is very similar to the previous proof (Theorem A.1). The difference is that the direction of movement not only affects one task's utility, but the utility difference comes from the execution of all tasks in the sequence. Of course, if a temporal constraint causes a task to start later than when the robot arrives there, there is no utility decrease due to deviating from the shortest route by moving in a different direction. For this reason, *critical* tasks are distinguished: a critical (non-delayed) task has no delay from time constraints compared to the corresponding schedule ($AP\,(\tau, s) > cstr(\tau)$ in Equation 3.14). Time constraints can be imposed from the search plan as detailed in Section 3.2.2. Because the delay in all the critical tasks in a schedule is the same as the delay in the first task, Equation A.5 changes as follows:

$$\frac{\partial E[U]}{\partial \mathbf{v}} * \frac{1}{\lambda} = \sum_i p_i * c_i \frac{\mathbf{v_i} \cdot \partial \mathbf{v}}{\partial \mathbf{v}}. \tag{A.11}$$

Here the sum is over all possible task outcomes. p_i, c_i and $\mathbf{v_i}$ stand for the chance of the i^{th} possibility, the number of critical task for the solution schedule, and the direction vector of the first task of the schedule. This will change the average direction in Equation A.10 to a weighted average with the weighting factors of the number of critical tasks for each schedule:

$$\arg\max_{\mathbf{v_a}} \frac{\partial E[U]}{\partial d} = \frac{\sum_i p_i c_i \mathbf{v_i}}{|\sum_i p_i c_i \mathbf{v_i}|}. \tag{A.12}$$

□

A.2 Solution Dominance and Alteration

In this section, we define solution dominance, and explain how a solution is altered towards dominant solutions in Chapters 3 and 4. A solution consists of a schedule to provide a solution to a given problem. A problem is a tuple that consists of a set of agents and a set of tasks ($P = \langle \mathbf{A}, \mathbf{T} \rangle$). Both agents and tasks are located in space. There is a distance measure defined between each two points that relates with how much time it takes to an agent to traverse from one location to another ($trav(s, \tau)$ in

Equation 3.12). Besides that, tasks have an activation time, only after which they can be executed ($cstr(\tau)$ in Equation 3.14).

A solution will consist of a schedule of tasks for each agent in the given problem ($S = \{s_a, \forall a \in \mathbf{A}\}$). There is an execution time ($t(\tau, S)$) defined for each task of the problem, that can be computed using the schedule, the traverse times, and the activation times according to Equation 3.12 and Equation 3.14.

Definition A.3. Solution dominance: Given two solutions to problem of MRTA with activation times for each task, S_a and S_b. S_a is weakly dominant over S_b ($S_a \succeq S_b$), if the execution time of every task according to schedule S_a is sooner or equal to according to S_b:

$$t(\tau, S_a) \leq t(\tau, S_b), \forall \tau \in \boldsymbol{T} \tag{A.13}$$

Definition A.4. Solutions with equal utility: when $S_a \succeq S_b$ and $S_a \preceq S_b$, then the two solutions have equal utilities ($S_a \sim S_b$):

$$t(\tau, S_a) = t(\tau, S_b), \forall \tau \in \boldsymbol{T} \tag{A.14}$$

In the following, an algorithm is presented that modifies a solution to another that dominates it (if possible). The modifications in the algorithm are split into atomic changes from two categories:

1. changes within a single schedule,
2. changes that affect multiple agent schedules.

In the following, the possible changes are detailed in each category.

A.2.1 Changes in a Single Schedule

The possible modifications within a single schedule mean that the same tasks are executed, but in a different order. The modifications are achieved by moving simple tasks within a schedule. We assume the triangle inequality hold for the task schedules, therefore the schedule of $[\tau_1, \tau_2]$ cannot be executed quicker than $[\tau_2]$ regardless of the location of τ_1:

$$t([s, \tau_1, \tau_2]) \geq t([s, \tau_2]) \tag{A.15}$$

Assuming this, any task can only be executed earlier if it is at an earlier point of the schedule. In order to find dominant solutions, each task is tried to be moved at an earlier point of the schedule, and if it results a dominant schedule (no later tasks are delayed because of it), then the change is kept and the rest of the tasks are tried the same way as detailed in Algorithm 4. Moreover, tasks that are not scheduled yet are also tried to fit at the end of the schedule, that also results in a dominant schedule.

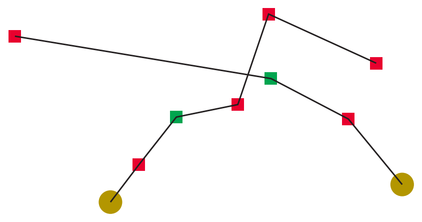
Algorithm 4 *singleAlter*: Single schedule alteration towards a dominant solution.

```
Require: $\mathbf{T}_u$: currently unscheduled tasks
Require: $s : \mathbb{N} \to \mathbf{T} \cup \{k | k = \varnothing\}$ schedule to modify
 1: for all $i \in \mathbb{N}, s(i) \neq \varnothing$ do
 2:     $\tau \leftarrow s(i)$
 3:     for all $j \in \mathbb{N}, j < i$ do
 4:         $s' \leftarrow insert(s \setminus \tau, \tau, j)$
 5:         if $s' \succeq s$ then
 6:             $s \leftarrow s'$
 7:         end if
 8:     end for
 9: end for
10: for all $\tau \in \mathbf{T}$ do
11:     if $\tau$ fits at the end of $s$ then
12:         $s(i) \leftarrow \tau$, where $i$ is smallest $s(i) = \varnothing$
13:         $\mathbf{T} \leftarrow \mathbf{T} \setminus \tau$
14:     end if
15: end for
Ensure: $s$
```

Here, $s \setminus \tau$ means schedule s with task τ removed.

A.2.2 Changes in Multiple Schedules

In order to perform changes that affect multiple agent schedules, tasks need to be exchanged between different agents. Such exchanges with minimal effect on the timing are when agents swap their remaining schedule after a certain task. This process is demonstrated in Figure A.2. The yellow circles represent agents, and rectangles represent task locations. In this example, the two agents swap their schedule after the tasks marked with green colour.


(a) Schedules before change.

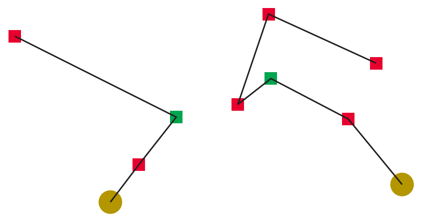

(b) Schedules after change.

Figure A.2: Schedule exchange example.

The exchange procedure can be broken down into atomic operations of one agent passes a part of its schedule to another agent. In order to achieve a successful procedure, these passes need to come back to the first agent, so the last agent needs to pass to the first agent. Each pass can be evaluated individually by its effect on solution dominance:

- *To dominant*(TD): Decreases the passed task execution times,
- *To equal*(TE): No change in the execution times,
- *To non-dominant*(TN): Increases the execution times of the passed tasks.

The possible actions can be represented on a directed graph, where the vertices represent the agents, and directed edges represent individual pass operations. In order to find an exchange that results in a dominant solution, a circle needs to be found with TD and TE edges.

Of course, also the task location after which the task schedule will be passed (insert index), needs to be determined. Fortunately, if a pass has a TN property, it will stay TN, if it is passed to a later point of the target agent schedule, because the execution times are consequently increasing in a schedule. This means, that increasing the insert index of other agents do not change a TN edge.

Now, if the insert index of each agent starts from the beginning, it is safe to increase the index of an agent if only TN edges are directed out of the vertex of this agent. It is because these edges will remain TN for any larger index of the other agents, and the vertex needs to have at least one TD or TE edge coming from it to be part of a circle of TD and TE edges, therefore it cannot be a part of an exchange that result in a dominant solution.

This means, that all agents insert index is increased if there are only TN edges from their vertex. In case none of the indices can be increased and haven't reached the end index, there is at least one TD or TE edge from each vertex. This means – assuming a finite graph – that there is a circle of only TD and TE edges in the graph. In order to avoid making the same exchange back and forth, at least one TD edge is required in order to perform an exchange. In case all the circles are made of only TE edges, a random vertex is chosen and its insert index is increased.

The above described exchange search process is formalised in Algorithm 5.

This algorithm terminates in finite time, because at each iteration at least one agent's insert index is increased, or a schedule exchange is performed. When the exchange is performed, the resulting solution is dominant (and not equal utility) to the original one because there is at least one TD edge. Due to the transitivity and irreversibility of the solution dominance excluding equal utility, and due to the finite number of solutions, only a finite number of exchange operations will be performed. Therefore, Algorithm 5

Algorithm 5 *multiAlter*: Multiple schedule alteration towards a dominant solution.

```
Require: $s : \mathbf{A} \times \mathbb{N} \to \mathbf{T} \cup \{k|k = \varnothing\}$: agent schedules
 1: $I : \mathbf{A} \to \mathbb{N}$: schedule indices
 2: $I(a) \leftarrow 1, \forall a \in \mathbf{A}$
 3: while $\exists s(a, I(a)) \neq \varnothing, a \in \mathbf{A}$ do
 4:     construct action graph
 5:     for all $a \in \mathbf{A}$ do
 6:         if only TN edges from vertex $a$ then
 7:             $I(a) \leftarrow I(a) + 1$
 8:         end if
 9:     end for
10:     $\mathbf{C} \leftarrow$ find circle with at least 1 TD edge and no TN edges
11:     if $\mathbf{C} = \emptyset$ then
12:         $a \leftarrow random\_choice(\{a : \forall a \in \mathbf{A}, s(a, I(a)) \neq \varnothing\})$
13:         $I(a) \leftarrow I(a) + 1$
14:     else
15:         $s \leftarrow exchange(s, I, \mathbf{C})$
16:     end if
17: end while
Ensure: $s$
```

terminates as there can only be a finite number of index increases and a finite number of exchange operations.

The *exchange* process is formalised in Algorithm 6.

Algorithm 6 *exchange*: Exchanging remaining schedules between agents along a circle.

```
Require: $s : \mathbf{A} \times \mathbb{N} \to \mathbf{T} \cup \{k|k = \varnothing\}$: agent schedules
Require: $I : \mathbf{A} \to \mathbb{N}$: schedule indices
Require: $\mathbf{C} = \{\langle a, b \rangle : a, b \in \mathbf{A}\}$: circle on action graph
 1: $s' : \mathbf{A} \times \mathbb{N} \to \mathbf{T} \cup \{k|k = \varnothing\}$
 2: for all $\langle a, b \rangle \in \mathbf{C}$ do
 3:     for all $i \in \mathbb{N}, i \leq I(b)$ do
 4:         $s'(b, i) \leftarrow s(b, i)$
 5:     end for
 6:     for all $i \in \mathbb{N}, i > I(a)$ do
 7:         $s'(b, i - (I(a) - I(b))) \leftarrow s(a, i)$
 8:     end for
 9: end for
10: for all $\langle a, b \rangle \in \mathbf{C}$ do
11:     $s(a, .) \leftarrow s'(a, .)$
12: end for
Ensure: $s$
```

Finally, the combination of these two methods is used to alter a solution to a dominant solution as detailed in Algorithm 7.

Algorithm 7 Random solution alteration towards a dominant solution.

```
Require: $s : \mathbf{A} \times \mathbb{N} \to \mathbf{T} \cup \{k|k = \varnothing\}$: agent schedules
 1: for all $a \in \mathbf{A}$ do
 2:     $s(a,.) \leftarrow singleAlter(s(a,.))$
 3: end for
 4: $s \leftarrow multiAlter(s)$
 5: while $s$ changes do
 6:     for all $a \in \mathbf{A}$ do
 7:         $s(a,.) \leftarrow singleAlter(s(a,.))$
 8:     end for
 9:     $s \leftarrow multiAlter(s)$
10: end while
Ensure: $s$
```

A.3 Bayesian Calculations for Solution Node Expansion

The problem encountered in Section 4.2.2.2 is determining the distribution of the mean next mean of a sample population after adding samples to it. The initial state is that there are n independent samples of a random variable, and we would like to add k further independent samples of the same distribution to it. In order to consider to make this action, the probability distribution of the new mean needs to be given.

At time t we have n samples of the random variable X: $x_1, x_2 ... x_n$ ($X \sim \mathcal{N}(\mu, \sigma)$). At time $t+1$ we add k further samples to it: $x_{n+1}, x_{n+2} ... x_{n+k}$. The sample means at time t and $t+1$ are the following:

$$\begin{aligned}\mu(t) = \mu_t &= \frac{\sum_{i=1}^{n} x_i}{n} \\ \mu(t+1) &= \frac{\sum_{i=1}^{n+k} x_i}{n+k}.\end{aligned} \tag{A.16}$$

The difference in these sample means will tell about the change in the mean value:

$$\mu(t+1) - \mu(t) = \frac{n * \mu_t + \sum_{i=n+1}^{n+k} x_i}{n+k} - \mu_t = \mu_t \left(\frac{n}{n+k} - 1\right) + \frac{\sum_{i=n+1}^{n+k} x_i}{n+k} \tag{A.17}$$

$$\mu(t+1) - \mu(t) \sim \mathcal{N}\left(\mu, \sqrt{\frac{\sigma^2}{n}}\right)\left(-\frac{k}{n+k}\right) + \mathcal{N}\left(\mu\frac{k}{n+k}, \frac{\sqrt{\sigma^2 k}}{n+k}\right) \tag{A.18}$$

$$\mu(t+1) - \mu(t) \sim \mathcal{N}\left(-\mu\frac{k}{n+k}, \frac{k}{n+k}\sqrt{\frac{\sigma^2}{n}}\right) + \mathcal{N}\left(\mu\frac{k}{n+k}, \frac{\sqrt{\sigma^2 k}}{n+k}\right) \tag{A.19}$$

$$\mu(t+1) - \mu(t) \sim \mathcal{N}\left(0, \sqrt{\frac{k^2}{(n+k)^2}\frac{\sigma^2}{n} + \frac{\sigma^2 k}{(n+k)^2}}\right) \tag{A.20}$$

$$\mu(t+1) - \mu(t) \sim \mathcal{N}\left(0, \sigma\frac{k}{n+k}\sqrt{\frac{1}{n} + \frac{1}{k}}\right) \tag{A.21}$$

To summarise, according to Equation A.21, the change in the mean of the samples ($Y = \mu(t+1) - \mu(t)$) is normally distributed with zero mean and $\sigma \frac{k}{n+k}\sqrt{\frac{1}{n}+\frac{1}{k}}$ standard deviation:

$$Y \sim \mathcal{N}\left(0, \sigma \frac{k}{n+k}\sqrt{\frac{1}{n}+\frac{1}{k}}\right). \tag{A.22}$$

Where σ is the standard deviation of the sampled random variable X, so far there are n samples drawn, and k additional samples are added at time $t+1$.

Appendix B

Simulation Framework Description

This appendix describes the code structure of the simulation frameworks used in this thesis.

B.1 C# framework for Chapters 3 and 4

In this section, we detail the operation of the planning approaches introduced in Chapters 3 and 4. The simulation framework is written in C#[1] for fast computation, easy debugging and flexible user interface.

The repository for the code of the simulation framework can be found at:
`https://bitbucket.org/zbeck/thesischapter3-4`.

B.1.1 Planners introduced in Chapter 3

The structure of the planning approaches in Chapter 3 can be seen in Figure B.1. The operation of the planners are covered in the `DualProblem` class. The individual elements of this class are the following:

- `Solve`: The root function to initiate the planning process.
- `SolveBottom`: Rescue planning using HOP (Algorithm 1).
- `SolveTopByFeedback`: Search planning using the rescue plans (Algorithm 2).
- `SolveTopSeparately`: Independent search planning (Section 3.2.1).

[1]`https://msdn.microsoft.com/en-us/library/a72418yk.aspx`

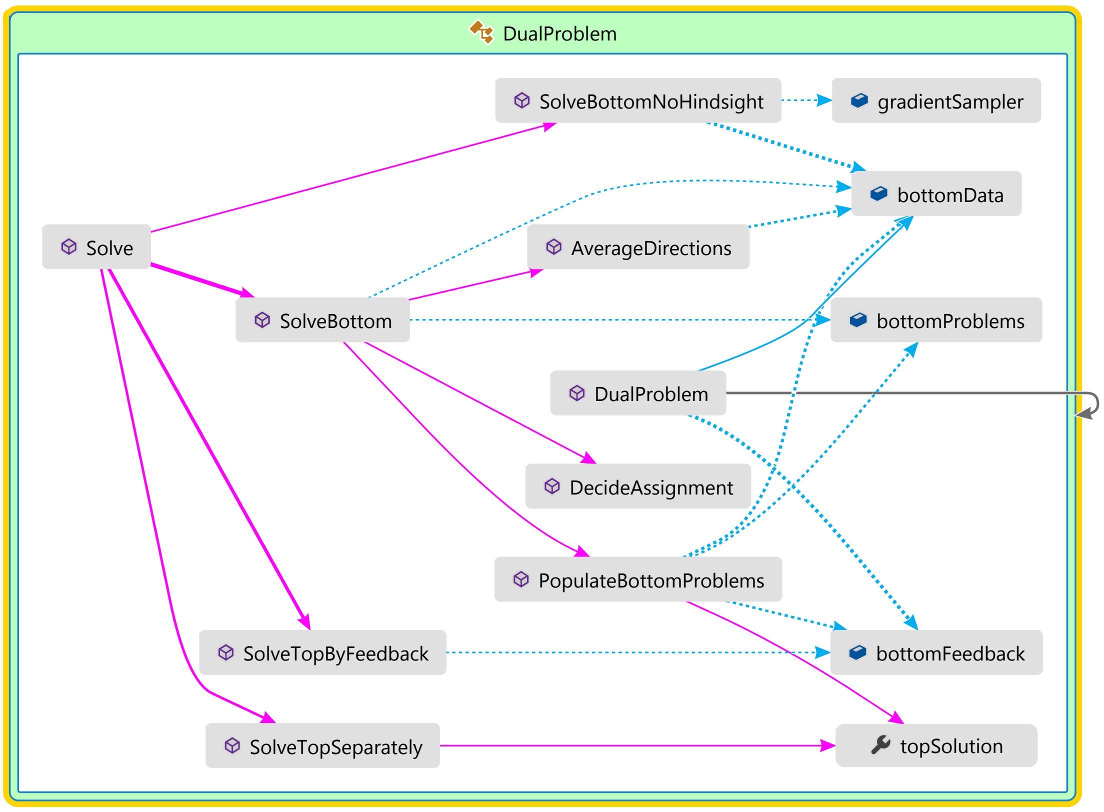


Figure B.1: Structure of the planning approaches introduced in Chapter 3

- `SolveBottomNoHindsight`: Rescue planning with gradient descent according to Section 3.2.1.1.
- `AverageDirections`: Responsible for Line 6 in Algorithm 1.
- `DualProblem`: Class constructor.
- `DecideAssignment`: Responsible for Lines 2, 3 and 4 in Algorithm 1.
- `PopulateBottomProblems`: Responsible for creating the hindsight problems and solutions for the rescue planning.
- `gradientSampler`: A class containing the gradients for the gradient descent algorithm.
- `bottomData`: Contains information about the rescue and helps creating hindsight problems.
- `bottomProblems`: The collection of hindsight problems.
- `bottomFeedback`: The rescue timings for search planning (**I** in Algorithm 2).
- `topSolution`: The latest plan for search.

B.1.2 MCTS introduced in Chapter 4

The structure of the planning approaches in Chapter 4 can be seen in Figure B.2. The operation of the MCTS planner is covered in the `MCSearch` class. Each node of the search tree extends the `ITreeNode` interface, and inherited from the `AbstractTreeNode` class. The problem nodes have `GroupProblem` type and the solution nodes have `GroupSolution` type. The main methods of the `ITreeNode` interface are the following:

- `Children`: Returns the list of children nodes.
- `Increase`: Finds the descendant with the highest expected utility increase from expansion.
- `IsBottomLevel`: If the child represents a terminal state (bottom level of the tree, rescue solution in our case).
- `Expand`: Expand the node with a number of children.

The individual elements of the `MCSearch` class are the following:

- `Solve`: The root function to initiate the planning process.
- `MCSearch`: Class constructor.
- `ExpandOnce`: Apply the *tree policy* of the MCTS (detailed in Section 2.2.4).
- `bottomData`: Contains information about the rescue and helps creating *rescue problem* nodes.
- `searchRoot`: The root node of the MCTS (in our case the *search problem*).
- `TimeStep`: The simulation time step when replanning is done.

B.2 Python framework for Chapter 5

The class diagram with the classes of the simulation framework used in Chapter 5 can be seen in Figure B.3. The most important methods of these classes are also highlighted. We provide a short description of these classes:

- `Commander` (*commander.py*): Responsible for creating and running the complete simulation.
- `CommFactory` (*agent.py*): A singleton class for handling broadcast requests and creating communicator instances.

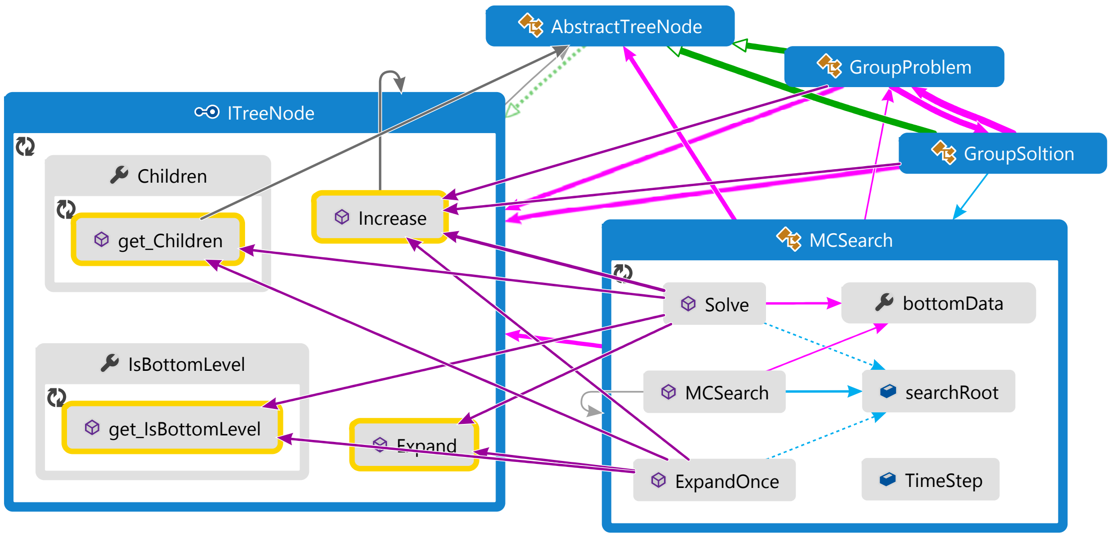


Figure B.2: Structure of the MCTS approach described in Chapter 4

- `AgentGroup` (*agentgroup.py*): Abstract class for a group of homogeneous agents.
- `SearchGroup` (*searchgroup.py*): Group of search agents. The method `run_agent_scherri_search()` and `run_agent_manvre()` are responsible for the two path-planning approaches (Section 5.3.2.1 and 5.3.2.2 respectively).
- `RescueGroup` (*rescuegroup.py*): Group of rescue agents.
- `NoSweepPlanner` (*sweep.py*): Route planner for fixed-wing UAVs passing through specific locations.
- `SensorSim` (*sensor_sim.py*): Simulates and models GSM signal detection (Section 5.2).
- `FixedWing` (*fixedsimulator.py*): Simulates a fixed-wing UAV's motion following waypoints in 2D.
- `Quad` (*quadsimulator.py*): Simulates a small rotary-wing UAV's motion following waypoints in 2D.
- `CommAgent` (*agent.py*): Abstract class for communicating agents able to receive messages of specific topics.
- `Searchr` (*Searchr.py*): A search planning agent.
- `ResQr` (*ResQr.py*): A rescue planning agent.
- `ManoeuvrePlanner` (*manvre.py*): An agent responsible for collecting, processing and providing information for short-term path planning for search (Section 5.3.2.2).
- `DeviateRouteSate` (*manvre.py*): A class used by a third-party MTCS planner for path planning purposes.

- `MultiTargetBeliefs` (*sensor_sim.py*): Maintains updates about mobile phones according to Section 5.2.
- `AgentState` (*agent.py*): A class for exchanging and preserving details for direct communication to agents.
- `SearchProblem` (*problem.py*): Contains a search MRTA problem and is able to solve it with the methods discussed in Section 3.2.
- `ProblemFactory` (*ProblemFactory.py*): Contains an UMRTA rescue problem and is able to provide HOP sample MRTA problems.
- `RescueProblem` (*problem_old.py*): Contains a hindsight rescue MRTA problem and is able to solve it with the methods discussed in Section 3.2.

The repository for the code of this simulation framework can be found at:
`https://bitbucket.org/zbeck/thesischapter5`.

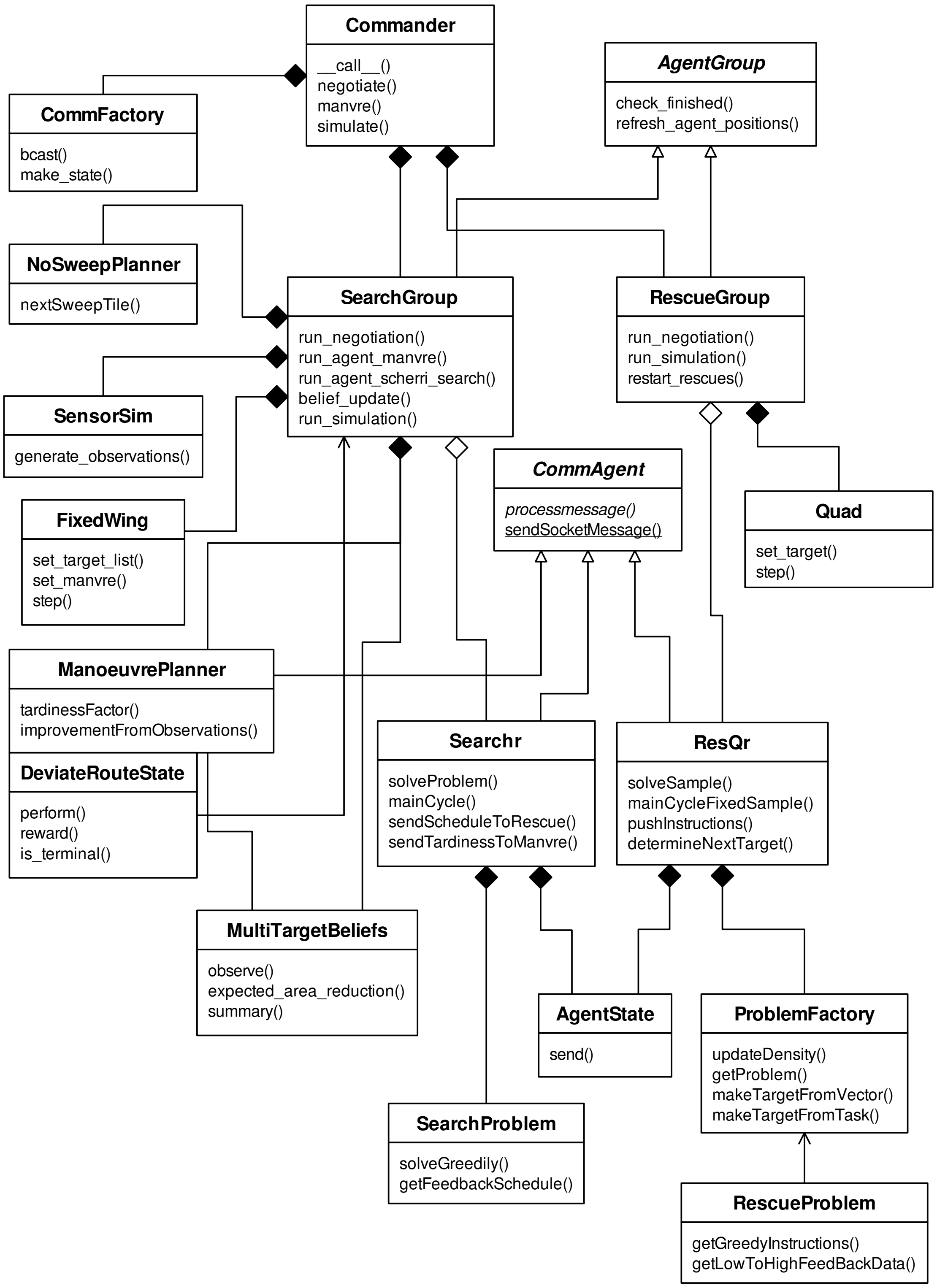


Figure B.3: Structure of the simulation framework described in Chapter 5

References

Akin, H. L., Ito, N., Jacoff, A., Kleiner, A., Pellenz, J., and Visser, A. (2013). Robocup rescue robot and simulation leagues. *The AI Magazine*, 34(1).

Amato, C. (2015). Cooperative decision making. In *Decision Making Under Uncertainty: Theory and Application*, pages 159–187. MIT Press.

Ambrus, A. (2011). Analysis of high resolution digital aerial imagery – the mapping of the sludge spill of october 2010 and statistical analysis. Master's thesis, Eötvös Loránd University, Faculty of Science. Bachelor Thesis.

Andriluka, M., Schnitzspan, P., Meyer, J., Kohlbrecher, S., Petersen, K., von Stryk, O., Roth, S., and Schiele, B. (2010). Vision based victim detection from unmanned aerial vehicles. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 1740–1747.

Andryeyev, O., Rubina, A., Golokolenko, O., Artemenko, O., and Mitschele-Thiel, A. (2016). SkySAIL: A flexible software-defined radio enabled micro aerial vehicle. In *2016 25th International Conference on Computer Communication and Networks (ICCCN)*, pages 1–6.

Angermann, M., Frassl, M., and Lichtenstern, M. (2012). Mission review of aerial robotic assessment – ammunition explosion cyprus 2011. In *Safety, Security, and Rescue Robotics (SSRR), 2012 IEEE International Symposium on*, pages 1–6.

Barbier, M., Cao, H., and Lacroix, S. (2009). Decision issues for multiple heterogeneous vehicles in uncertain environments. In *Control Architectures of Robots, 4th National Conference on.*

Baxter, J. L., Burke, E. K., Garibaldi, J. M., and Norman, M. (2009). Shared potential fields and their place in a multi-robot co-ordination taxonomy. *Robotics and Autonomous Systems*, 57(10):1048 – 1055. 5th International Conference on Computational Intelligence, Robotics and Autonomous Systems (5th CIRAS).

Beck, Z., Teacy, W. L., Rogers, A., and Jennings, N. R. (2016). Online planning for collaborative search and rescue by heterogeneous robot teams. In *AAMAS 16: 15th Int. Conf. on Autonomous Agents and Multi-Agent Systems*, pages 1024–1033. International Foundation for Autonomous Agents and Multiagent Systems.

Belbachir, A., Lacroix, S., Ingrand, F., Perrier, M., and Opderbecke, J. (2010). Cooperative-adaptive algorithms for targets localization in underwater environment. In *Autonomous Underwater Vehicles (AUV), 2010 IEEE/OES*, pages 1–7.

Bellingham, J., Tillerson, M., Richards, A., and How, J. (2003). Multi-task allocation and path planning for cooperating UAVs. In Butenko, S., Murphey, R., and Pardalos, P., editors, *Cooperative Control: Models, Applications and Algorithms*, volume 1 of *Cooperative Systems*, pages 23–41. Springer US.

Bernardini, S., Fox, M., Long, D., and Bookless, J. (2013). Autonomous search and tracking via temporal planning. In *Proceedings of the 23st International Conference on Automated Planning and Scheduling (ICAPS-13).*

Birkmann, J., editor (2013). *Measuring Vulnerability to Natural Hazards: Towards Disaster Resilient Societies.* United Nations University Press.

Bjarnason, R., Fern, A., and Tadepalli, P. (2009). Lower bounding klondike solitaire with Monte-Carlo planning. In *Proceedings of the 19th International Conference on Automated PLanning and Scheduling*, pages 26–33.

Boumghar, R. and Lacroix, S. (2011). Over the hill and far away: aerial/ground cooperation for long range navigation. In *Proceedings of the 14th International Conference on Climbing and Walking Robots and the Support Technologies for Mobile Machines, Paris (France)*, pages 215–222. World Scientific.

Browne, C., Powley, E., Whitehouse, D., Lucas, S., Cowling, P., Rohlfshagen, P., Tavener, S., Perez, D., Samothrakis, S., and Colton, S. (2012). A survey of Monte Carlo tree search methods. *Computational Intelligence and AI in Games, IEEE Transactions on*, 4(1):1–43.

Brutschy, A., Tran, N.-L., Baiboun, N., Frison, M., Pini, G., Roli, A., Dorigo, M., and Birattari, M. (2012). Costs and benefits of behavioral specialization. *Robotics and Autonomous Systems*, 60(11):1408 – 1420. Towards Autonomous Robotic Systems 2011.

Burgard, W., Moors, M., Fox, D., Simmons, R., and Thrun, S. (2000). Collaborative multi-robot exploration. In *Robotics and Automation, 2000. Proceedings. ICRA '00. IEEE International Conference on*, volume 1, pages 476–481 vol.1.

Burns, E., Benton, J., Ruml, W., Yoon, S., and Do, M. (2012). Anticipatory on-line planning. In *International Conference on Automated Planning and Scheduling.*

Bush, L. A., Williams, B., and Roy, N. (2008). Computing exploration policies via closed-form least-squares value iteration. In *Doctoral Consortium, Eighteenth International Conference on Automated Planning & Scheduling (ICAPS-08).*

Cameron, S., Hailes, S., Julier, S., McClean, S., Parr, G., Trigoni, N., Ahmed, M., McPhillips, G., De Nardi, R., Nie, J., et al. (2010). SUAAVE: Combining aerial robots and wireless networking. *Unmanned Air Vehicle Systems, Bristol*, 1865:7–20.

Cao, H., Lacroix, S., Ingrand, F., and Alami, R. (2010). Complex tasks allocation for multi robot teams under communication constraints. In *5th National Conference on "Control Architectures of Robots" Douai (France).*

Carlson, J. and Murphy, R. (2005). How UGVs physically fail in the field. *Robotics, IEEE Transactions on*, 21(3):423–437.

Chaimowicz, L. and Kumar, V. (2004). A framework for the scalable control of swarms of unmanned ground vehicles with unmanned aerial vehicles. In *In Proceedings of the 10th International Conference on Robotics and Remote Systems for Hazardous Environments.*

Chandler, P., Pachter, M., Nygard, K., and Swaroop, D. (2002). Cooperative control for target classification. In Murphey, R. and Pardalos, P., editors, *Cooperative Control and Optimization*, volume 66 of *Applied Optimization*, pages 1–19. Springer US.

Chao, I.-M., Golden, B. L., and Wasil, E. A. (1996). The team orienteering problem. *European Journal of Operational Research*, 88(3):464 – 474.

Chaslot, G. M. J. B., Winands, M. H. M., and van den Herik, H. J. (2008). Parallel Monte-Carlo tree search. In van den Herik, H. J., Xu, X., Ma, Z., and Winands, M. H. M., editors, *Computers and Games: 6th International Conference, CG 2008, Beijing, China, September 29 - October 1, 2008. Proceedings*, pages 60–71. Springer Berlin Heidelberg, Berlin, Heidelberg.

Chen, J. Y. C., Barnes, M. J., and Harper-Sciarini, M. (2011). Supervisory control of multiple robots: Human-performance issues and user-interface design. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 41(4):435–454.

Chung, C. F. and Furukawa, T. (2009). Coordinated pursuer control using particle filters for autonomous search-and-capture. *Robotics and Autonomous Systems*, 57(6–7):700 – 711.

Chung, T. and Burdick, J. (2008). Multi-agent probabilistic search in a sequential decision-theoretic framework. In *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, pages 146–151.

Coburn, A., Pomonis, A., Sakai, S., and Spence, R. (1991). Assessing human casualties caused by building collapse in earthquakes. In *Int. Conf. on the Impact of Natural Disasters, USA.*

Coles, A. I., Long, D., and Rendell, P. (2010). Experiences with temporal planning. In *Proceedings of the Workshop of the UK Planning and Scheduling Special Interest Group (PlanSIG)*.

Cowling, P. I., Powley, E. J., and Whitehouse, D. (2012). Information set Monte Carlo tree search. *IEEE Transactions on Computational Intelligence and AI in Games*, 4(2):120–143.

Crispin, C. and Sobester, A. (2015). An intelligent, heuristic path planner for multiple agent unmanned air systems. In *AIAA SciTech 56th AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*. American Institute of Aeronautics and Astronautics.

DARPA Robotics Challenge (2013). Official webpage. `http://www.theroboticschallenge.org/`.

De Ville de Goyet, C., Grünewald, F., and Sarmiento, J. P. (2011). Health responses to the earthquake in haiti january 2010: Lessons to be learned for the next massive sudden-onset disaster. Technical report, Pan American Health Organization (PAHO).

Doherty, P. and Rudol, P. (2007). A UAV search and rescue scenario with human body detection and geolocalization. In Orgun, M. and Thornton, J., editors, *AI 2007: Advances in Artificial Intelligence*, volume 4830 of *Lecture Notes in Computer Science*, pages 1–13. Springer Berlin Heidelberg.

Dubins, L. E. (1957). On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents. *American Journal of mathematics*, 79(3):497–516.

Edlinger, R., Zauner, M., and Rokitansky, W. (2013). Intelligent mobility - new approach of robot mobility systems for rescue scenarios. In *Safety, Security, and Rescue Robotics (SSRR), 2013 IEEE International Symposium on*, pages 1–5.

Erginer, B. and Altug, E. (2007). Modeling and PD control of a quadrotor VTOL vehicle. In *2007 IEEE Intelligent Vehicles Symposium*, pages 894–899.

Farinelli, A., Rogers, A., Petcu, A., and Jennings, N. R. (2008). Decentralised coordination of low-power embedded devices using the max-sum algorithm. In *Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems - Volume 2*, AAMAS '08, pages 639–646, Richland, SC. International Foundation for Autonomous Agents and Multiagent Systems.

Fave, F. M. D., Farinelli, A., Rogers, A., and Jennings, N. (2012). A methodology for deploying the max-sum algorithm and a case study on unmanned aerial vehicles. In *IAAI 2012: The Twenty-Fourth Innovative Applications of Artificial Intelligence Conference*, pages 2275–2280.

Ferworn, A., Tran, J., Ufkes, A., and D'Souza, A. (2011). Initial experiments on 3D modeling of complex disaster environments using unmanned aerial vehicles. In *Safety, Security, and Rescue Robotics (SSRR), 2011 IEEE International Symposium on*, pages 167–171.

Fiedrich, F. and Burghardt, P. (2007). Agent-based systems for disaster management. *Commun. ACM*, 50(3):41–42.

Fischer, J., Jiang, W., Kerne, A., Greenhalgh, C., Ramchurn, S. D., Reece, S., Pantidi, N., and Rodden, T. (2014). Supporting team coordination on the ground: Requirements from a mixed reality game. In *11th Int. Conference on the Design of Cooperative Systems (COOP 2014)*.

Fox, M. and Long, D. (2011). PDDL2.1: An Extension to PDDL for Expressing Temporal Planning Domains. *ArXiv e-prints*.

Furukawa, T., Durrant-Whyte, H., Dissanayake, G., and Sukkarieh, S. (2003). The coordination of multiple UAVs for engaging multiple targets in a time-optimal manner. In *Intelligent Robots and Systems, 2003. (IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on*, volume 1, pages 36–41 vol.1.

Gan, S. K., Fitch, R., and Sukkarieh, S. (2012). Real-time decentralized search with inter-agent collision avoidance. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 504–510.

Gan, S. K. and Sukkarieh, S. (2011). Multi-UAV target search using explicit decentralized gradient-based negotiation. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 751–756.

Gasparini, L., Norman, T. J., and Kollingbaum, M. J. (2016). Observation-based multi-agent planning with communication. In *ECAI 2016: 22nd European Conference on Artificial Intelligence, 29 August-2 September 2016, The Hague, The Netherlands-Including Prestigious Applications of Artificial Intelligence (PAIS 2016)*, volume 285, page 444. IOS Press.

Gerkey, B. and Mataric, M. (2002). Sold!: auction methods for multirobot coordination. *Robotics and Automation, IEEE Transactions on*, 18(5):758–768.

Gerkey, B. and Mataric, M. (2003). Multi-robot task allocation: analyzing the complexity and optimality of key architectures. In *Robotics and Automation, 2003. Proceedings. ICRA '03. IEEE International Conference on*, volume 3, pages 3862–3868 vol.3.

Gerkey, B. P. and Matarić, M. J. (2004). A formal analysis and taxonomy of task allocation in multi-robot systems. *The International Journal of Robotics Research*, 23(9):939–954.

Goodrich, M., Cooper, J., Adams, J., Humphrey, C., Zeeman, R., and Buss, B. (2007). Using a mini-UAV to support wilderness search and rescue: Practices for human-robot teaming. In *Safety, Security and Rescue Robotics, 2007. SSRR 2007. IEEE International Workshop on*, pages 1–6.

Goodrich, M. A., Morse, B. S., Gerhardt, D., Cooper, J. L., Quigley, M., Adams, J. A., and Humphrey, C. (2008). Supporting wilderness search and rescue using a camera-equipped mini UAV. *Journal of Field Robotics*, 25(1-2):89–110.

Grath, F. M. (2014). Mobile network restoration & humanitarian response. Technical report, GSMA.

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

Guy Carpenter (2010). Toxic industrial spill, Ajka, Hungary. `http://gcportal.guycarp.com/wcportalapp-gca/publicsite/catdocument.pdf?instratreportid=1983`.

Han, J., Xu, Y., Di, L., and Chen, Y. (2013). Low-cost multi-UAV technologies for contour mapping of nuclear radiation field. *Journal of Intelligent & Robotic Systems*, 70(1-4):401–410.

Hansen, P., Mladenović, N., and Moreno Pérez, J. A. (2010). Variable neighbourhood search: methods and applications. *Annals of Operations Research*, 175(1):367–407.

Hart, P. E., Nilsson, N. J., and Raphael, B. (1968). A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100–107.

Hatazaki, K., Konyo, M., Isaki, K., Tadokoro, S., and Takemura, F. (2007). Active scope camera for urban search and rescue. In *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2596–2602.

Hespanha, J. P., Kim, H. J., and Sastry, S. (1999). Multiple-agent probabilistic pursuit-evasion games. In *Decision and Control, 1999. Proceedings of the 38th IEEE Conference on*, volume 3, pages 2432–2437 vol.3.

Jing, D., Jian, C., and Min, S. (2009). Cooperative task assignment for heterogeneous multi-UAVs based on differential evolution algorithm. In *Intelligent Computing and Intelligent Systems, 2009. ICIS 2009. IEEE International Conference on*, volume 2, pages 163–167.

Johnson, N., Kotz, S., and Balakrishnan, N. (1995). *Continuous univariate distributions*. Number v. 2 in Wiley series in probability and mathematical statistics: Applied probability and statistics. Wiley & Sons.

Jones, D. (2014a). Exercise angel thunder. *Crisis Response*, 10:30–31.

Jones, D. (2014b). Human-agent collectives. *Crisis Response*, 10:62–63.

Katrasnik, J., Pernus, F., and Likar, B. (2010). A survey of mobile robots for distribution power line inspection. *IEEE Transactions on Power Delivery*, 25(1):485–493.

Kawata, Y. (1995). The great hanshin-awaji earthquake disaster : Damage, social response, and recovery. *Journal of natural disaster science*, 17(2):1–12.

Kingman, J. F. C. (1992). *Poisson processes*, volume 3. Oxford university press.

Kocsis, L. and Szepesvári, C. (2006). Bandit based Monte-Carlo planning. In Fürnkranz, J., Scheffer, T., and Spiliopoulou, M., editors, *Machine Learning: ECML 2006*, volume 4212 of *Lecture Notes in Computer Science*, pages 282–293. Springer Berlin Heidelberg.

Kullback, S. and Leibler, R. A. (1951). A probabilistic approach to concurrent mapping and localization for mobile robots. *Annals of Mathematical Statistics*, 22(1):79–86.

Kvarnström, J. (2010). Planning for loosely coupled agents using partial order forward-chaining. In *The Swedish AI Society Workshop 2010, SAIS 2010 :*, number 48 in Linköping Electronic Conference Proceedings, pages 45–54. Linköping University Electronic Press, Linköpings universitet.

Lavalle, S. M. (1998). Rapidly-exploring random trees: A new tool for path planning. Technical report, Citeseer.

Le, T., Norman, T. J., and Vasconcelos, W. (2009). Agent-based sensor-mission assignment for tasks sharing assets. In *Proceeding of the Third International Workshop on Agent Technology for Sensor Networks*, pages 33–40.

Leary, S., Deittert, M., and Bookless, J. (2011). Constrained UAV mission planning: A comparison of approaches. In *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on*, pages 2002–2009.

Leetaru, K. (2015). How drones are changing humanitarian disaster response. `http://www.forbes.com/sites/kalevleetaru/2015/11/09/how-drones-are-changing-humanitarian-disaster-response`.

Lemaire, T., Alami, R., and Lacroix, S. (2004). A distributed tasks allocation scheme in multi-UAV context. In *Robotics and Automation, 2004. Proceedings. ICRA '04. 2004 IEEE International Conference on*, volume 4, pages 3622–3627 Vol.4.

Levi, N., Kovelman, G., Geynis, A., Sintov, A., and Shapiro, A. (2013). The DARPA virtual robotics challenge experience. In *Safety, Security, and Rescue Robotics (SSRR), 2013 IEEE International Symposium on*, pages 1–6.

Liu, Y. and Nejat, G. (2013). Robotic urban search and rescue: A survey from the control perspective. *Journal of Intelligent & Robotic Systems*, 72(2):147–165.

Liu, Y., Yang, J., Zheng, Y., Wu, Z., and Yao, M. (2013). Multi-robot coordination in complex environment with task and communication constraints. *International Journal of Advanced Robotic Systems*, 10(229).

Lubrano, E. (2013). Drone adventure in haiti. `http://www.droneadventures.org/2013/05/29/haiti/`.

Luo, C., Espinosa, A., Pranantha, D., and De Gloria, A. (2011). Multi-robot search and rescue team. In *Safety, Security, and Rescue Robotics (SSRR), 2011 IEEE International Symposium on*, pages 296–301.

Luotsinen, L. J., Gonzalez, A. J., Bölöni, L., and Orlando, C. F. (2004). Collaborative UAV exploration of hostile environments. *Proceedings for the Army Science Confetence (24th).*

Macarthur, K., Stranders, R., Ramchurn, S., and Jennings, N. (2011). A distributed anytime algorithm for dynamic task allocation in multi-agent systems. In *Twenty-Fifth Conference on Artificial Intelligence (AAAI)*, pages 701–706. AAAI Press. Event Dates: August 7-11, 2011.

MapAction (2010). Haiti: Earthquake – SAR competed sectors port-au-prince. `http://mapaction.org/component/mapcat/mapdetail/1971.html`.

Marconi, L., Leutenegger, S., Lynen, S., Burri, M., Naldi, R., and Melchiorri, C. (2013). Ground and aerial robots as an aid to alpine search and rescue: Initial SHERPA outcomes. In *Safety, Security, and Rescue Robotics (SSRR), 2013 IEEE International Symposium on*, pages 1–2.

Martínez-de Dios, J., Merino, L., Ollero, A., Ribeiro, L., and Viegas, X. (2007). Multi-UAV experiments: Application to forest fires. In Ollero, A. and Maza, I., editors, *Multiple Heterogeneous Unmanned Aerial Vehicles*, volume 37 of *Springer Tracts in Advanced Robotics*, pages 207–228. Springer Berlin Heidelberg.

Masato, D., Norman, T. J., and Vasconcelos, W. W. (2009). Agent support for human team collaboration in uncertain environments. In *Proc. of the First Int. Workshop on Mixed-Initiative Multiagent Systems, Budapest, Hungary.*

Mathews, G. M., Durrant-Whyte, H., and Prokopenko, M. (2009). Decentralised decision making in heterogeneous teams using anonymous optimisation. *Robotics and Autonomous Systems*, 57(3):310 – 320. Selected papers from 2006 IEEE International Conference on Multisensor Fusion and Integration (MFI 2006) 2006 IEEE International Conference on Multisensor Fusion and Integration.

Meier, P. (2013). How UAVs are making a difference in disaster response. `http://irevolution.net/2013/12/05/uavs-in-disaster-response/`.

Michael, N., Shen, S., Mohta, K., Mulgaonkar, Y., Kumar, V., Nagatani, K., Okada, Y., Kiribayashi, S., Otake, K., Yoshida, K., Ohno, K., Takeuchi, E., and Tadokoro, S. (2012). Collaborative mapping of an earthquake-damaged building via ground and aerial robots. *Journal of Field Robotics*, 29(5):832–841.

Monahan, G. E. (1982). State of the art—a survey of partially observable markov decision processes: Theory, models, and algorithms. *Management Science*, 28(1):1–16.

Moorehead, S., Simmons, R., and Whittaker, W. (2001). Autonomous exploration using multiple sources of information. In *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*, volume 3, pages 3098–3103 vol.3.

Morrow, N., Mock, N., Papendieck, A., and Kocmich, N. (2011). Independent evaluation of the ushahidi haiti project. *Development Information Systems International*, 8.

Mulero-Pázmány, M., Stolper, R., van Essen, L. D., Negro, J. J., and Sassen, T. (2014). Remotely piloted aircraft systems as a rhinoceros anti-poaching tool in africa. *PLoS ONE*, 9(1):1–10.

Munoz-Castaner, J., Soto, P. C., Gil-Castineira, F., Gonzalez-Castano, F. J., Ballesteros, I., di Giovanni, A., and Villar, P. C. (2015). Your phone as a personal emergency beacon: A portable GSM base station to locate lost persons. *IEEE Industrial Electronics Magazine*, 9(4):49–57.

Murphy, R. (2011). The 100:100 challenge for computing in rescue robotics. In *Safety, Security, and Rescue Robotics (SSRR), 2011 IEEE International Symposium on*, pages 72–75.

Nagatani, K., Kiribayashi, S., Okada, Y., Otake, K., Yoshida, K., Tadokoro, S., Nishimura, T., Yoshida, T., Koyanagi, E., Fukushima, M., and Kawatsuma, S. (2013). Emergency response to the nuclear accident at the Fukushima Daiichi Nuclear Power Plants using mobile rescue robots. *Journal of Field Robotics*, 30(1):44–63.

Nakanishi, A. (2016). “119-Ban de ‘kyūmei dorōn’, kusuri ya AED todokeru… konshū ni jisshō jikken” １１９番で「救命ドローン」、薬やＡＥＤ届ける…今秋に実証実験 [“Lifesaving drone” delivers medicine and AED when you call 119… experiments this fall]. *Yomiuri Online*, News commentary.

Nanjanath, M. and Gini, M. (2010). Repeated auctions for robust task execution by a robot team. *Robotics and Autonomous Systems*, 58(7):900 – 909. Advances in Autonomous Robots for Service and Entertainment.

Nielsen, J. (1993). *Usability Engineering.* Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

OCHA (2014). Unmanned aerial vehicles in humanitarian response. In *OCHA Policy and Studies Series.* United Nations Office for the Coordination of Humanitarian Affairs.

Ogren, P., Fiorelli, E., and Leonard, N. (2004). Cooperative control of mobile sensor networks: Adaptive gradient climbing in a distributed environment. *Automatic Control, IEEE Transactions on*, 49(8):1292–1302.

Olfati-Saber, R. (2006). Flocking for multi-agent dynamic systems: algorithms and theory. *Automatic Control, IEEE Transactions on*, 51(3):401–420.

Pastor-Escuredo, D., Morales-Guzmán, A., Torres-Fernández, Y., Bauer, J. M., Wadhwa, A., Castro-Correa, C., Romanoff, L., Lee, J. G., Rutherford, A., Frias-Martinez, V., Oliver, N., Frias-Martinez, E., and Luengo-Oroz, M. (2014). Flooding through the lens of mobile phone activity. In *Global Humanitarian Technology Conference (GHTC), 2014 IEEE*, pages 279–286.

Perkins, T. and Murphy, R. (2013). Active and mediated opportunistic cooperation between an unmanned aerial vehicle and an unmanned ground vehicle. In *Safety, Security, and Rescue Robotics (SSRR), 2013 IEEE International Symposium on*, pages 1–8.

Pesaresi, M., Kemper, T., Gueguen, L., and Soille, P. (2010). Automatic information retrieval from meter and sub-meter resolution satellite image data in support to crisis management. In *Geoscience and Remote Sensing Symposium (IGARSS), 2010 IEEE International*, pages 1792–1795.

Police of Hungary (2010). Spill simulation model reproducing the sludge spill at the ajka alumina plant. Simulated using the OSIRIS system of the Police of Hungary, using the DTA-50 and DDM-50 databases of the Hungarian Defese Froce. `http://index.hu/belfold/2010/10/08/modelleztek_a_tobbi_zagytarozot/`.

Pujol-Gonzalez, M., Cerquides, J., Meseguer, P., Rodríguez-Aguilar, J., and Tambe, M. (2013). Engineering the decentralized coordination of UAVs with limited communication range. In Bielza, C., Salmerón, A., Alonso-Betanzos, A., Hidalgo, J., Martínez, L., Troncoso, A., Corchado, E., and Corchado, J., editors, *Advances in Artificial Intelligence*, volume 8109 of *Lecture Notes in Computer Science*, pages 199–208. Springer Berlin Heidelberg.

Punch, M. and Markham, G. (2000). Policing disasters: the british experience. *Int'l J. Police Sci. & Mgmt.*, 3:40.

Qi, J., Song, D., Shang, H., Wang, N., Hua, C., Wu, C., Qi, X., and Han, J. (2016). Search and rescue rotary-wing UAV and its application to the lushan ms 7.0 earthquake. *Journal of Field Robotics*, 33(3):290–321.

Rabadi, G., Anagnostopoulos, G. C., and Mollaghasemi, M. (2007). A heuristic algorithm for the just-in-time single machine scheduling problem with setups: a comparison with simulated annealing. *The International Journal of Advanced Manufacturing Technology*, 32(3):326–335.

Rabadi, G., Mollaghasemi, M., and Anagnostopoulos, G. C. (2004). A branch-and-bound algorithm for the early/tardy machine scheduling problem with a common due-date and sequence-dependent setup time. *Computers & Operations Research*, 31(10):1727 – 1751.

Radhakrishnan, S. and Ventura, J. A. (2000). Simulated annealing for parallel machine scheduling with earliness-tardiness penalties and sequence-dependent set-up times. *International Journal of Production Research*, 38(10):2233–2252.

Ramchurn, S., Fischer, J., Ikuno, Y., Wu, F., Flann, J., and Waldock, A. (2015). A study of human-agent collaboration for multi-UAV task allocation in dynamic environments. In *International Joint Conference on Artificial Intelligence.*

Ramchurn, S. D., Wu, F., Jiang, W., Fischer, J. E., Reece, S., Roberts, S., Rodden, T., Greenhalgh, C., and Jennings, N. R. (2016). Human–agent collaboration for disaster response. *Autonomous Agents and Multi-Agent Systems*, 30(1):82–111.

Rappaport, T. S. et al. (1996). *Wireless communications: principles and practice*, volume 2. Prentice Hall PTR New Jersey.

Rasche, C., Stern, C., Richert, W., Kleinjohann, L., and Kleinjohann, B. (2010). Combining autonomous exploration, goal-oriented coordination and task allocation in multi-UAV scenarios. In *Autonomic and Autonomous Systems (ICAS), 2010 Sixth International Conference on*, pages 52–57.

Rescue Global (2013). Disaster response. Technical report, Rescue Global.

Scerri, P., Glinton, R., Owens, S., Scerri, D., and Sycara, K. (2007). Geolocation of RF emitters by many UAVs. In *"AIAA InfotechAerospace 2007 Conference and Exhibit"*. American Institute of Aeronautics and Astronautics.

Scerri, P., Von Gonten, T., Fudge, G., Owens, S., and Sycara, K. (2008). Transitioning multiagent technology to UAV applications. In *Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems: Industrial Track*, AAMAS '08, pages 89–96, Richland, SC. International Foundation for Autonomous Agents and Multiagent Systems.

Schesvold, D., Tang, J., Ahmed, B. M., Altenburg, K., and Nygard, K. E. (2003). POMDP planning for high level UAV decisions: Search vs. strike. In *In Proceedings of the 16th International Conference on Computer Applications in Industry and Engineering.*

SHERPA (2013). Sherpa deliverable d2.1 – benchmark definition, related performance measures & platform specifications. Technical report, SHERPA Project.

SHERPA (2014). Sherpa deliverable d2.2 – application scenarios, end user requirements and regulations. Technical report, SHERPA Project.

Smith, R. G. (1980). The contract net protocol: High-level communication and control in a distributed problem solver. *Computers, IEEE Transactions on*, C-29(12):1104–1113.

Spry, S., Girard, A., and Hedrick, J. (2005). Convoy protection using multiple unmanned aerial vehicles: organization and coordination. In *American Control Conference, 2005. Proceedings of the 2005*, pages 3524–3529 vol. 5.

Sukhatme, G. S., Montgomery, J. F., and Vaughan, R. T. (2001). Experiments with cooperative aerial-ground robots. In *Robot Teams: From Diversity to Polymorphism. AK Peters*, pages 345–367.

Syouryuu, K. (2016). "Kumamoto kyōkun ni jishin taiō toku-ka, Dazaifu Chikushino ryōshi no bōsai kunren" 熊本教訓に地震対応特化、太宰府・筑紫野両市の防災訓練 [Special training after the Kumamoto earthquake, disaster response exercise in Dazaifu and Chikushino cities]. *Yomiuri Online*, Kyushu departure news.

Tadokoro, S. (2009). Earthquake disaster and expectation for robotics. In Tadokoro, S., editor, *Rescue Robotics*, pages 1–16. Springer London.

Tressel, P., Boon, F., Kohli, S., Koenig, D., Lopez, B., Lev, E., Howden, M., and Goldenberg, A. (2014). *Sahana Eden.* Lulu.com.

United Nations Foundation, Harvard Humanitarian Initiative, and OCHA (2010). Disaster relief 2.0: The future of information sharing in humanitarian emergencies. Technical report, HHI; United Nations Foundation; OCHA; The Vodafone Foundation.

UNOSAT (2010). Haiti earthquake 2010: Remote sensing based building damage assessment data. `http://www.unitar.org/unosat/haiti-earthquake-2010-remote-sensing-based-building-damage-assessment-data`.

UNOSAT (2011). Unosat humanitarian rapid mapping service. `http://unosat.web.cern.ch/unosat/unitar/publications/Overview2011UNOSATRapidMapping_final2.pdf`.

USCG (1995). Search and rescue – chapter 9. In *Model Maritime Service Code*. United States Coast Guard.

van der Horst, J. and Noble, J. (2010). Distributed and centralized task allocation: When and where to use them. In *Self-Adaptive and Self-Organizing Systems Workshop (SASOW), 2010 Fourth IEEE International Conference on*, pages 1–8.

Vansteenwegen, P., Souffriau, W., and Oudheusden, D. V. (2011). The orienteering problem: A survey. *European Journal of Operational Research*, 209(1):1 – 10.

Waharte, S. and Trigoni, N. (2010). Supporting search and rescue operations with UAVs. In *Emerging Security Technologies (EST), 2010 International Conference on*, pages 142–147.

Wood, L., Buscher, M., van Veelen, B., and Van Splunter, S. (2012). Agile response and collaborative agile workflows. In *Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE), 2012 IEEE 21st International Workshop on*, pages 358–363.

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

Yokoo, M. (2012). *Distributed Constraint Satisfaction: Foundations of Cooperation in Multi-agent Systems*. Springer Publishing Company, Incorporated, 1st edition.

Yoon, S., Fern, A., Givan, R., and Kambhampati, S. (2008). Probabilistic planning via determinization in hindsight. In *Proceedings of the 23rd National Conference on Artificial Intelligence - Volume 2*, AAAI'08, pages 1010–1016. AAAI Press.

Zivan, R. and Sycara, K. (2010). Cooperation between search and surveillance agents in DCOP_MST. In *Workshop 18 The Twelfth International Workshop on Distributed Constraint Reasoning DCR 2010*.