Factored Monte-Carlo Tree Search for Coordinating UAVs in Disaster Response

Chris A. B. Baker, Sarvapali Ramchurn, W. T. Luke Teacy, Nicholas R. Jennings
Agents Interactions and Complexity Group, University of Southampton
Southampton SO17 1BJ, United Kingdom
{cabb1g08, sdr1, wtlt}@soton.ac.uk, n.jennings@imperial.ac.uk

Abstract

The coordination of multiple Unmanned Aerial Vehicles (UAVs) to carry out surveys is a major challenge for emergency responders. In particular, UAVs have to fly over kilometre-scale areas while trying to discover casualties as quickly as possible. However, an increase in the availability of real-time data about a disaster from sources such as crowd reports or satellites presents a valuable source of information to drive the planning of UAV flight paths over a space in order to discover people who are in danger. Nevertheless challenges remain when planning over the very large action spaces that result. To this end, we introduce the survivor discovery problem and present as our solution, the first example of a factored coordinated Monte Carlo tree search algorithm to perform decentralised path planning for multiple coordinated UAVs. Our evaluation against standard benchmarks show that our algorithm, Co-MCTS, is able to find more casualties faster than standard approaches by 10% or more on simulations with real-world data from the 2010 Haiti earthquake.

Introduction

The increased prevalence of low-cost, robust, commercially available Unmanned Aerial Vehicles (UAVs) has led to concerted efforts to utilise these platforms in disaster response. Specifically, UAVs have seen widespread use in recent disasters—such as the 2010 Haiti earthquake and the 2015 Nepal earthquake—aiding first responders with collecting imagery and other sensory data without putting human lives at risk (Adams and Friedland 2012; Goda et al. 2015; Meier 2015). In particular, an important body of work has focused on developing UAVs that act as *autonomous systems* to minimise the involvement of overstretched first responders: both to conserve valuable manpower and to ensure their safety (Crisis Mappers 2013; Murphy 2012; United Nations Foundation 2011). Key to this work, is the idea of enabling coordinated UAVs to explore a disaster space to discover the spatial location of casualties: a difficult task given the extent and large number of possible locations to visit.

To enable this exploration, advances in data collection—and specifically crowd-sourcing (Morrow et al. 2011; Goodchild and Glennon 2010)—have created new sources of information about disaster scenarios that contribute to increased awareness of the situation on the ground during a disaster. Such information is vital given the size and scale of the effects of natural disasters (Fowler 2016). For instance, during 2010 a magnitude 7.3 earthquake struck Haiti near the capital, Port-Au-Prince, which caused widespread destruction over thirty kilometres from the epicentre and resulted in the destruction or damage of over three hundred thousand homes; the death or injury of over five hundred and twenty thousand people; and around one point three million displaced persons needing temporary shelter (Government of the Republic of Haiti 2014). A natural extension of the use of UAVs in these events is to fully exploit prior data on the extent and nature of the disaster (whether crowd-sourced or otherwise) in order to maximise the likelihood of rescuing or discovering casualties quickly. However, at present there is no specific work that seeks to use spatial information on the distribution of *people* and the expected *danger*—for instance the likely rate of fatalities due to structural damage to buildings or from the spread of radiation—to inform the paths of UAVs through a disaster space, in order to maximise the number of observations made of possible casualties.

Currently, the state of the art for UAV path planning algorithms focuses on three main areas, the first of which is target tracking for surveillance (Bernardini, Fox, and Long 2014; Hu et al. 2014; Kolling and Kleiner 2013). Now, although these techniques are related to the exploration of a disaster space, they are designed to find a known number of targets that are in motion, rather than an unknown number of survivors distributed over an area. Other developments in path planning focus on trying to reach a set goal location (or state) (Chen et al. 2014; Durkota and Komenda 2013; He 2007; Kothari, Postlethwaite, and Gu 2009) or working with single autonomous UAVs (Cashmore et al. 2014; Kothari and Postlethwaite 2012); neither of which fulfil the need for algorithms that coordinate multiple vehicles in an *explorative* traversal of the disaster space, rather than aiming for a particular final location. Specific challenges exist in coordinating multiple vehicles. For example, there is often no benefit to multiple UAVs providing imagery of the same location: there must be coordination between the vehicles to allow them to find survivors in a disaster, without all attending the same locations.

In line with the terminology used in this field (Bry and Roy (2011), Gan and Sukkarieh (2011), Waharte, Trigoni, and Julier (2009) and others) we henceforth refer to data

distributed over a spatial representation of a disaster area as a *belief map*. This belief map can come from different sources, including crowd-sourcing, but we assume it has the characteristics of mapping spatial locations onto some function that represents numerical data: for instance number of people or radiation levels.

Our work seeks to address these challenges with the following three contributions:

1. We introduce a formulation of the *survivor discovery* problem, informed by several datasets collected following natural disasters;
2. We develop a decentralised algorithm that allows multiple UAVs to coordinate the exploration of a large disaster space with many states;
3. We test and evaluate our approach on real-world data simulating a very large action space, showing consistent gains in survivor discovery of over 10% compared to benchmarks.

The rest of the paper is organised as follows. First we discuss the background of using UAVs in disaster response and the types of problems that existing research has focussed on; next we describe the specifics of the problem we seek to solve by formulating the survivor discovery problem; we then detail our use of a coordinated Monte-Carlo tree-search algorithm (including examples); before showing the benefits of our implementation with three experimental scenarios using real-world datasets; and finally summarising our work and proposing an extension of the algorithm and a removal of some assumptions.

Background

In order to best use UAVs to aid responders in disasters they must be able to plan paths autonomously, as a group. Furthermore, as we have already indicated, it is beneficial to use prior information about the area to inform the flight paths of UAVs in order to maximise the likelihood of discovering survivors. Currently, work on path planning in robotics focusses primarily on reaching goal locations and frequently formulates path planning as a control problem (Goerzen, Kong, and Mettler 2009). Conversely, in a disaster scenario there need not be any final end-point to a UAV's path planning; rather the length of the exploration may be constrained by—for example—battery life, and the number of people to be discovered must be maximised over the length of the path. Alternatively, much work has also been done to enable the use of vision algorithms and belief data to track mobile targets or map an area (Liu and Dai 2010). However, this area of research often focusses on locating a known number of targets, or covering a bounded space for the purposes of mapping. In contrast, the task of searching for casualties in disaster response can be summarised as the localisation of an *unknown* number of people as quickly as possible (Fawcett and Oliveira 2000; Chiu et al. 2002), in order to ensure quick rescue or attention by emergency services and thus the greatest chance of survival (Macintyre, Barbera, and Petinaux 2011). Consequently techniques for known numbers of targets or for mapping environments are not useful.

Against this background, we find closer similarities with work on solving Markov Decision Processes (MDPs); specifically where locality of UAVs can be used to reduce calculation overheads. In particular, the generality of the MDP formulation lends itself well to the construction of a simulation environment given numerical data used for a belief map, as well as having a number of well-established solutions. Specifically, work by Amato and Oliehoek (2015) utilises factored tree-searches for partially observable MDP solutions, and demonstrates performance benefits over state of the art un-factored solver POMCP (Silver and Veness 2010). This performance advantage is obtained by exploiting problem structure to allow factorisation in a way reflective of local state spaces and interactions. In a similar way, we use factored trees in this paper to represent the available actions of UAVs in a disaster environment, factoring the value of locating people between UAVs within spatial proximity of each other. Notably—because of the dimensions of the physical area we consider—we deal with a very large state-space, which must be particularly carefully sampled since complete searches are computationally intractable. We also include—for the first time—real data from a disaster scenario to validate our model.

We next introduce the specific formulation of the problem we tackle, and explain the origin of the large state-space.

The Survivor Discovery Problem

In this section to formulate the problem of finding a number of survivors in a disaster-area, with a team of cooperative UAVs planning their actions and coordinating with each other if required, in order to maximise the number of people discovered. All of this is to be done in the presence of a model of the danger to the casualties expressed as an estimated fatality rate mapped to spatial locations. As such the problem is one of prioritising visits to areas of high expected numbers of people, while also attending areas with high expected fatality rates as quickly as possible. We now describe; in turn; the environment model, the behaviour of UAVs in the simulation, and the specific problem of discovering survivors. Following this we introduce the Bellman equation (used to characterise MDPs) for our scenario.

Environment Model

We begin by discretising the search-space to allow for fast plan creation. Specifically, we formulate the survivor discovery problem as exploring a uniform $x \times y$ sized grid world—C—formed of cells,[1] $c_{ij} \in C$. Each cell c_{ij} contains an *unknown* number of people, $p_{ij} \in \mathbb{Z}^*$, and a scalar value

$$d_{ij} \in [0, 1] \tag{1}$$

denoting the *danger* in the cell as the probability of *any* person in the cell dying during the next time step. Time steps are denoted by an integer value $t \in \mathbb{Z}^*$, with subsequent steps referred to by adding integer values; for example $t+1$. If a value depends on time, this is denoted in parentheses; i.e. $p_{ij}\,(t)$.

[1] Cells are indexed for their horizontal and vertical position respectively by ij.

UAV Behaviour Formulation

The area C is explored by the set of UAVs $U = \{u_1, \ldots, u_m\}$ that traverse C from cell to cell once per timestep. Each UAV is equipped with sensors capable of accurately detecting people inside one cell at any given timestep: i.e. they return the value of p_{ij} deterministically (this is a simplifying assumption we will address in further work). Each UAV, u_k, must select its own trajectory $T_k \subseteq C$ with contiguous borders, through the area, forming the set of all UAV trajectories $\mathbf{T} = \{T_1, \ldots, T_m\}$. To address double-counting the values in each cell,[2] we introduce the union of all the trajectories in the set $\mathbf{T}$ as $\mathcal{T}(\mathbf{T}) = \bigcup_{T_k \in \mathbf{T}} T_k$. Additionally, we impose constraints on the length of each UAV's planned trajectory to account for limits on battery life. If the maximum number of cells that can be traversed due to the battery limitations of a UAV u_k is denoted $b_k \in \mathbb{Z}^+$, then the maximum trajectory length is simply $\mid T_k \mid \leq b_k \forall k \in \{1, \ldots, m\}$; which we denote f.

The action vectors enabling the UAVs to transition from cell to cell are defined as $a_k = (\leftarrow, \rightarrow, \uparrow, \downarrow)$ for each UAV u_k, (each arrow representing the direction of motion: we do not consider the UAVs remaining stationary as according to our model this can never result in more reward than moving). We impose the constraint that the available actions are restricted to UAVs at the edge of the grid world (i.e. explicitly where the UAV occupies a cell c_{ij} where $i = 0$ or x, or $j = 0$ or y) so that they do not have the action available to cross out of the grid area. At any given time, the vector denoting all possible UAV actions is given by $a = (a_1, \ldots, a_m)$, and represents all possible combinations of movement available to all UAVs. The set of all actions available to all UAVs at any arbitrary time forms the *total* action space A, which we now incorporate into the standard Bellman equation.

Exploration Problem Formulation

Having described the environment and UAV behaviour, we formulate the multi-UAV exploration problem as a Markov Decision Process (MDP), comprising a tuple $\langle S, A, R, P \rangle$ of states S, actions A, rewards R, and transition probabilities P, which we define below. Since the UAVs are not aware in advance of the ground-truth values of p_{ij} and d_{ij} in each cell, computations are instead made using prior belief-data about the expected number of people $\bar{p}_{ij} \in \mathbb{R}^*$ and the expectation value of the probability of death $\bar{d}_{ij} \in [0, 1]$ of each person in a cell in a given time step. We also consider a binary variable $v_{ij} \in \{0, 1\}$ denoting the *visibility* of the cell: i.e. whether it has been observed by a UAV. This allows us to construct the tuple s_{ij} to denote the state of a cell c_{ij}:

$$s_{ij} = \langle \bar{p}_{ij}, \bar{d}_{ij}, v_{ij,} \rangle$$

The set of these for all cells in C forms the global state variable $s = \{s_{00}, \ldots, s_{xy}\}$. With this constructed, we next formulate an update procedure for the expected value of the number of people in a given cell after a time step t by computing the product of the current expected number of people and the probability of survival:

$$\bar{p}_{ij}(t+1) = \bar{p}_{ij}(t)(1 - \bar{d}_{ij}) \tag{2}$$

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

$$\bar{d}_{ij}(t+1) = \begin{cases} 0 & if\ v_{ij}\,(t) = 1 \\ \bar{d}_{ij}\,(t) & otherwise \end{cases}$$

This assumption need not hold in general. However, it is a convenient way of indicating that no further utility can be derived from revisiting a cell once it has been observed. In a scenario with trapped survivors it is reasonable to assume there will be no large-scale movement of population during the search. Indeed, in many cases victims are often recovered after being trapped in the same location for days at a time (Macintyre, Barbera, and Petinaux 2011).

We note the logical extension of equation 2 to times at an arbitrary point in the future t', as required for planning future actions:

$$\bar{p}_{ij}\,(t') = \bar{p}_{ij}\,(t)\left(1 - \bar{d}_{ij}\right)^{t'-t} \tag{3}$$

We record the set of positions of each UAV $u_k \in U$ at time t as $g_k\,(t) \in C$, which we denote as members of the vector of all UAV positions:

$$g(t) = (g_1(t), \ldots, g_m(t))$$

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

$$G\,(t) = \bigcup_{k=1}^{m} \{g_k\,(t)\}$$

Thus, the state of the map at a time t is a tuple comprising the state of each cell, and the position of each UAV: $\tilde{s}\,(t) = \langle s\,(t)\,, g\,(t) \rangle$.

As a result, we formulate the immediate reward to all UAVs at timestep t as a function of the expected number of people saved due to UAV observation. Specifically, this is the product of the expected number of people in a cell multiplied by the death rate in that cell, summed over all unique cells where a UAV is present (i.e. all members of the set G):

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

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

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

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

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

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

Thus, R must be maximised over the trajectory of *all* UAVs in order to observe (and subsequently 'save') the maximum number of people. The key challenge is to allow the path planning to be coordinated between UAVs to maximise global, rather than local, reward (shown explicitly by the inclusion of G in Equation). We briefly discuss the implications of the UAVs detecting p_{ij} with certainty, and justify this assumption, at the end of this section.

Bellman Equation

Having formulated the environment, the UAVs, and the discovery problem; we finally formulate the standard MDP Bellman optimality equation, where we denote R's dependence on the actions and state space (which themselves depend on t):

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

In our case, we do not require the diminishing-returns term γ to ensure $Q\left(a,s\right)$ is finite, since the diminishing value of examining a cell further in the future is expressed in the reduction of the expected number of people as a result of the death-rate term (see Equation 2); which is incorporated in the reward function. In other words, $Q\left(a,s\right)$ cannot exceed the number of people alive in the space, given s.

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

$$Q(a,s) = R\left(a,s\right) + \max_{a'} Q'\left(a',s'\right) \tag{4}$$

from which we obtain the optimal action: $a = \arg\max_a Q$ to maximise current and future reward. While ostensibly a simplification of a typical MDP formulation, the problem is far from trivial as the joint action space a at any time step grows as an exponent of the number m of UAVs in the system, namely: $\|a\| \propto a_k^m$, while the possible combinations of G grow proportional to:

$$\mid C \mid! = (x \times y)! \tag{5}$$

i.e. a combinatorial function of the dimensions of the environment. The result is a state space S of extremely large size (we give a specific example in our Results section below) even before accounting for the permutations of p, d, and v.

We now briefly discuss the implications of the discoveries in the environment happening deterministically and how this does not affect our planning.

Determinism in our Model

With regards to the rewards obtained in our formulation, we note here the implications of determinism of detection of people in our model. Whilst the construction above relies on expected values of reward for exploration of a cell—since we cannot in advance know the true conditions on the ground (implied above in the unknown quantities p_{ij} and d_{ij})—we can still consider our MDP model deterministic, with the following justification. All planning in our decision making processes can only rely on the expected value of cell reward. Once a cell has been visited, and the true reward discovered, further exploration of the cell in our model yields no further reward (see below). Furthermore, at this stage we do not consider correlation between adjacent cells. This is because we ensure the decomposition results in cells encompassing entire buildings that in Haiti—and in Port au Prince in particular—are often built sporadically and have little relation to the structures in their surroundings (Government of the Republic of Haiti 2014). As a result, planning is not affected upon discovery of the true reward of visiting a cell, and we can therefore consider the prediction of future expected reward deterministic. Having clarified this point, we now introduce our solution to the survivor discovery problem in the environment model just described.

The Coordinated Monte Carlo Tree-Search Algorithm

Our decision to base our algorithm on Monte-Carlo tree search (MCTS) methods is due to their ability to sample very quickly from large state spaces (traditionally used in solving games), and the flexibility with which they can be applied to general problems (including MDPs) (Browne et al. 2012; Amato and Oliehoek 2015). This former point is particularly vital in our scenario as we have already noted the extremely large number of configurations possible in the environment (Equation 5). The principal purpose of our algorithm is to allow UAVs to use MCTS algorithms to calculate coordinated paths without incurring the cost described in Equation 5. To do this we exploit locality between UAVs to factor the search space into local joint-action trees. Furthermore, we allow trees to coordinate over shared factors (that is, shared UAVs). To this end we use the max-sum algorithm (Ramchurn et al. 2010; Rogers et al. 2011) as we note its ability to guarantee neighbourhood maximal rewards, and can do so in relatively few iterations (Farinelli, Rogers, and Jennings 2014).

Specifically, we introduce an additional step to the standard MCTS process of tree growth. This growth is typically summarised: node *selection*, *expansion*, *rollout* or simulation, and *backpropagation* (Browne et al. 2012; Kocsis and Szepesvari 2006). However, in our case we need to allow the UAVs to search their future actions whilst also accounting for the actions of other UAVs and the impact they will have on the reward obtained (that is, on the survivors discovered). Most significantly, we modify the selection process to determine which node to expand by coordinating in parallel between trees via max-sum: resulting in an exploration algorithm factored between multiple subsets of UAVs. This represents the first time a factored tree-search has been applied to a UAV search simulation. We detail our approach in the following subsections.

Tree Construction

At each timestep in the simulation, the coordinated MCTS (Co-MCTS) algorithm begins by calculating which UAVs

require coordination with their neighbours, leading to the form of the UAV-based factor graph constructed in the joint-action creation function $\mathcal{J}$ (Line 3). This is performed to establish whether coordination is needed in a given UAV's locality. In cases where a UAV is spatially isolated from neighbouring UAVs, a local tree is grown. The resulting groups of UAVs will form the basis of the factor graph used in the max-sum calculation (Line 14 in Algorithm 1). The result of $\mathcal{J}$ is represented formally by a set $N = \{n_1, \ldots, n_f\}$ that represents the domain of the factor nodes to be coordinated. Specifically, each member of N contains a set of actions corresponding to a group of UAVs that require coordination.

In more detail, for some set of neighbouring UAVs—for example $\{u_1, u_2, \ldots, u_k\}$—the possibility exists of $g_1(t+1) = g_2(t+1) = \ldots = g_k(t+1)$ at the next time step $t+1$ of a simulation.[3] In this case, the corresponding element in N—say n_i—would be the set of actions available to these UAVs: $n_i = \{a_1, a_2, \ldots, a_k\}$. Notice that since a UAV may interact (that is, potentially occupy the same cell at a future timestep, and thus form a joint tree) with more than one neighbour, the condition $\bigcup_{n_k \in N} n_k = G$ must hold, whereas $\bigcap_{n_k \in N} n_k = \emptyset$ will, in general, not. Trees are grown for each n_i in N, each of which in turn represents the factors in the max-sum graph connected to the variables representing the available actions of the UAVs. Individual nodes in the tree n_i will be indicated as $n_i^{(k)}$ or from any arbitrary tree by $n^{(k)}$.

Having described the construction of the trees, we now detail how each tree is grown in order to sample from the action space.

Tree Growth

Algorithm 1 begins with the creation of the root nodes representative of each factor seen in Line 6, which are recorded in the set N_r (Line 4). Following this, the creation and growth of branches is performed Δ times inside the loop beginning at Line 9. This begins by exploring down each tree, starting from the root node, to determine which node to branch on next. Similarly to standard upper confidence bound MCTS (Kocsis and Szepesvari 2006), this begins by selecting nodes which have hitherto not been fully expanded: that is, there remain neighbouring action states that have not yet been branched to previously. The total number of neighbouring actions ν from a given action node is of order $\nu = \prod_{a_\beta \in n_\alpha} \mid a_\beta \mid$ for a tree corresponding to factor node n_α, simplifying to $\nu = \mid a_\beta \mid^{|n_\alpha|}$ when all UAVs in the set have the same number of available actions.[4] Thus, the operation of the function in Line 18:

$$\text{fullexp}(n^{(k)}) = \begin{cases} True & \text{if } e = \nu \\ False & \text{otherwise} \end{cases}$$

where e is the number of previous expansions of that node. Line 10 introduces the current set of nodes (across all trees) to be expanded next, N_{next}, and Line 11 creates the set of previously expanded nodes N_{prev}. At Line 14 the max-sum algorithm is used to maximise the value of the actions over each n_i, returning a vector of favourable actions $a^* = (a_1^*, a_2^*, \ldots, a_m^* \mid a_k^* \in a_k)$. Since each n_i depends on a subset of actions, the function select$(n^{(k)}, a^*)$ serves to return only the actions corresponding to a given $n^{(k)}$. This is then used as the argument to create the new expansion to a node in N_{next} in Line 17.

Algorithm 1 Coordinated MCTS

```
CoMCTS(G, C, t = 0)
1.  for each in [1, ..., f]
2.    //Creation of factor graphs//
3.    N ← J(G)
4.    N_r ← ∅
5.    for n_i in N
6.      append(N_r) ← n_i^(0)
7.    endfor
8.    //Loop for each iteration of tree growth//
9.    for δ in [1, ..., Δ]
10.     N_next ← N_r
11.     N_prev ← ∅
12.     while N_next ≠ ∅
13.       //Coordination between trees for production of optimal actions//
14.       a* ← maxsum (N, N_next)
15.       for n^(k) in N_next
16.         //Chooses action relevant to the tree//
17.         n_new^(k) ← expand (n^(k), select (n^(k), a*))
18.         if fullexp (n^(k)) = True
19.           remove (n^(k), N_next)
20.           append (n^(k), N_prev)
21.         endif
22.       endfor
23.     endwhile
24.     for n^(k) in N_prev
25.       //Rollout and backpropagation of values//
26.       rollout(n_new^(k))
27.       backpropagate(n_new^(k))
28.   endfor
29.   endfor
30.   t ← t + 1
31. endfor
32. for n_i in N
33.   Return (bestactions (n_i))
34. endfor
```

An example factor graph and trees are shown in Figure 1, for four interacting UAVs. Their actions are shared between two factor graph utility nodes, hence the two joint action trees n_1 and n_2. Four expansions of the root are shown where the action of the shared UAV—u_3—has been coordinated between the trees each time, thus ensuring contradictory actions are not chosen for the same UAV in two different trees. A second depth of growth is shown in Figure 2, where coordination has resulted in a newly created node with common action for a_3 of $\uparrow$.

Rollout

The rollout portion of the MCTS is traditionally a coarse estimate of the affect of future actions as the result of explor-

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

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

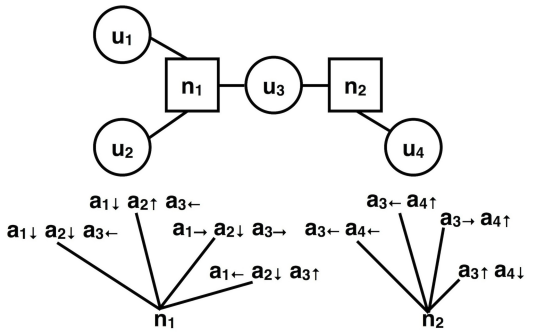

Figure 1: Example factor graph and the first iteration of tree growth for four interacting UAVs. Here u_3 can interact with any of the other UAVs, and is thus common to the two joint-action trees, which coordinate in order to maximise the reward from its actions in conjunction with the other UAVs. Factor and utility nodes are synonymous with variable and function nodes as outlined in (Ramchurn et al. 2010).

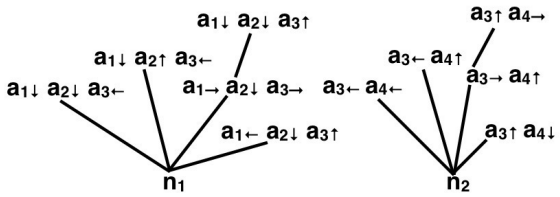

Figure 2: New nodes created at lower depth in the tree, where the actions of u_3 have again been coordinated to be $\uparrow$ in both cases.

ing a particular node in the action space. In this example, we base the rollout on a random-walk through the action space starting at the node just expanded, biased in the direction of the last action taken. This method has the benefit of showing not just the contribution of any random series of actions, but of taking more actions similar to the one represented by the frontier node (for each UAV). Intuitively, a random rollout from one node in a joint action tree will be insignificantly different from a rollout from any similar node because of their spatial proximity. Conversely, our rollout policy contributes to the exploration value of a node by indicating possible future reward through continued tree expansion with a preference for repetitions of the action itself.

Results

To verify the performance of our algorithm on data relevant to real-world disaster scenarios, we used data from the Ushahidi project (Morrow et al. 2011) produced from crowd-sourced information during the 2010 Haiti earthquake.[5] Specifically, we extracted the level of damage and coordinates of buildings in a 2km square centred on the capital, Port-au-Prince. Damage was rated based on crowd reports on a scale from 1 to 5, with 5 being the most severe.

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

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

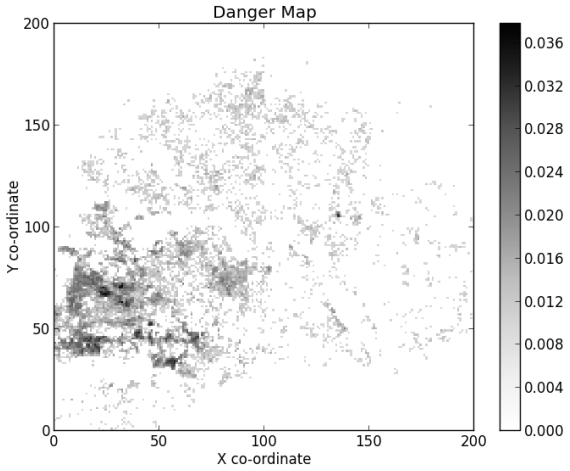

Figure 3: Danger as a function of position, created from Ushahidi dataset centred over Port-au-Prince

The number of unique sets of UAV positions G (that is, the cardinality of the state space) in this environment for (for example) five UAVs, is of the order 8.5×10^{20} (referring to Equation 5); even without the variation in the expected number of people with time.[6] Using this environment, we measure performance by evaluating the number of survivors averaged over each UAV over each timestep. That is: $\frac{1}{f \cdot m} \sum_{t=0}^{f} \sum_{c_{ij} \in G(t)} \bar{p}_{ij}(t)$ where f denotes the final time step (a stopping time introduced simply as a limit of the simulation), and we recall that m is the total number of UAVS, and $G(t)$ is the unique positions of the UAVs at time step t.

Figure 4 shows our results for an initial test of the coordinated Monte Carlo tree search using randomised starting locations for four simulated UAVs. We compare against, a simple "lawnmower" sweep search—a typical strategy employed in search and rescue situations (Goodrich et al.

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

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

2007)—as well as a typical MCTS algorithm (as per the work of Chaslot et al. (2008)) without the factored coordination we introduce. We also demonstrate explicitly the forward planning required in the survivor discovery problem, by benchmarking against Co-MCTS without a rollout policy, and a greedy (but locally coordinated via max-sum) policy. Results demonstrate a minimum performance increase of 10% over MCTS, and higher gains over the other benchmarks. Errors are taken as standard error of the mean over 1000 repeats of each experiment. The poor performance of a greedy one-step lookahead shows that even with coordination, the ability to forward-plan to account for survivor death rates is essential to discovering casualties in our scenario. Indeed, simple un-planned lawnmower-style sweep searches are more successful, but still fall short of the MCTS's ability to plan exploration paths to target (for instance) high-danger areas before casualty rates become too high.

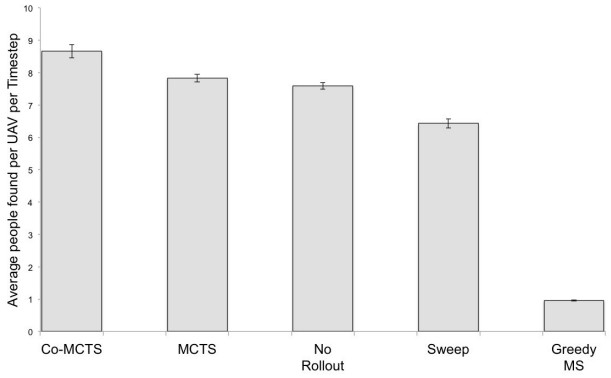

Figure 4: Comparison of performance between: coordinated MCTS, un-coordinated MCTS, simple lawnmower sweep search, Co-MCTS with no rollout policy, greedy policy with max-sum coordination. Start locations were uniformly randomised, $m = 4$, $f = 1000$.

Additionally, we explicitly demonstrate the benefit of the *consistency* afforded by our coordination in a simulation. We do this by varying the number of UAVs present and comparing to the closest benchmark to Co-MCTS's performance from our initial simulation. Good coordination should benefit the overall reward gained by the algorithm (in our case the number of people rescued) with low diminishing returns compared to uncoordinated approaches. Specifically, any additional UAVs should still find close-to the same number of casualties as other UAVs in the system, if they coordinate the exploration task effectively as a group. If they do not, one would expect additional UAVs would explore the same regions of the disaster space as those already present: which, as discussed previously, offers no benefit to the global reward function. This is demonstrated in Figure 5 for varying m, where we note performance improvements of 12% with the inclusion of 6 UAVs (and an average of 6% across the experiment). This is notable after 3 UAVs are introduced, since before this point interaction (and therefore required coordination between the UAVs) was at a minimum since the search space was large enough to accommodate multiple non-intersecting explorative paths. For more UAVs, coordination is more commonplace and with the successful implementation of Co-MCTS there is higher value in the inclusion of further UAVs into the scenario, compared to a situation without coordination.

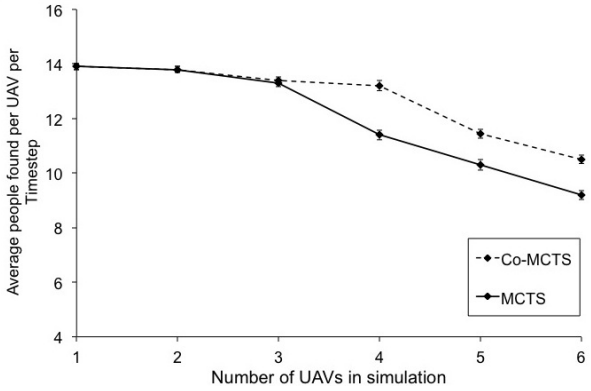

Figure 5: Comparison of coordinated and un-coordinated MCTS in locating survivors with additional UAVs; demonstrating a consistent performance per-UAV in Co-MCTS. Initial UAV starting locations fixed at $c_{0,0} = [0, 0]$; $f = 1000$.

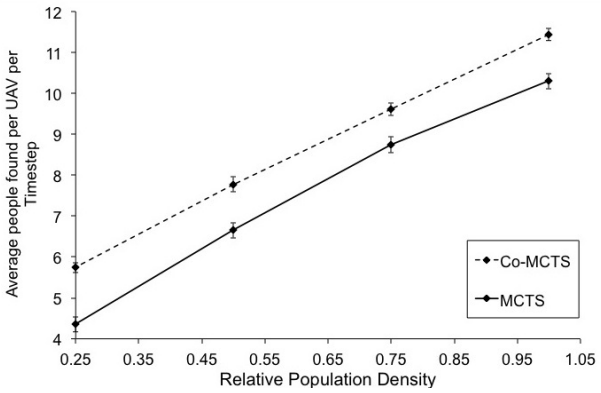

Figure 6: Comparison of coordinated and un-coordinated MCTS in locating survivors over varying densities of population, showing Co-MCTS's consistency in different forms of belief data. Here $c_{0,0} = [0, 0]$; $f = 1000$; and $m = 5$.

Finally, we demonstrate how Co-MCTS performs consistently in varying environments. For the algorithm to be viable, it should show performance benefits in a variety of situations with a variety of distributions of casualties to be discovered. We achieve this by varying the relative population over the belief-space from 1 (as above), down to 0.25 by uniformly sampling from the Ushahidi dataset to the population portion of the belief map. This has the effect of maintaining our standard of using real data with a large action space, while still allowing us to experiment over different state spaces (S). The results presented in Figure 6 demonstrate this consistency with an average 14% improvement over the un-coordinated MCTS benchmark: even at low densities of people; where the coordination requirements would

be intuitively less valuable. We thus provide evidence that the performance gains of Co-MCTS are not simply a by-product of the test environment selected.

Conclusions

Motivated by the recent increased availability of belief-data about disaster environments, we have introduced an implementation of a decentralised, factored, coordinated Monte Carlo tree search algorithm for the purpose of discovering survivors in a simulated UAV path planning scenario. Tests were carried out on real-world data from the 2010 Haiti earthquake via the Ushahidi platform; an environment with a very large (of order 8.5×10^{20}) action space. We demonstrated the capability of our Co-MCTS algorithm in sampling this space and planning paths, and demonstrated consistent performance gains in the number of survivors discovered of $> 10\%$. Future work will seek to extend these solutions to time-varying belief maps to cope with the situation in which data is collected and updated during exploration, and removing the assumption of a cellular disaster area to tackle the issue of discretising (and therefore potentially oversimplifying) the data collected.

References

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

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

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

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

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

Cashmore, M.; Fox, M.; Larkworthy, T.; Long, D.; and Magazzeni, D. 2014. AUV Mission Control via Temporal Planning. In *2014 IEEE International Conference on Robotics and Automation*, 6535–6541. Hong Kong, China: IEEE.

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

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

Chiu, W.-t.; Arnold, J.; Shih, Y.-T.; Hsiung, K.-H.; Chi, H.-Y.; Chiu, C.-H.; Tsai, W.-C.; and Huang, W. C. 2002. A Survey of International Urban Search-and-rescue Teams following the Ji Ji Earthquake. *Disasters* 26(1):85–94.

Crisis Mappers. 2013. Crisis Mappers - The Humanitarian Technology Network.

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

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

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

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

Fowler, J. 2016. Data curbs earthquake risk in Armenia. Technical report, United Nations Office for Disaster Risk Reduction.

Gan, S. K., and Sukkarieh, S. 2011. Multi-UAV Target Search using Explicit Decentralized Gradient-Based Negotiation. In *2011 IEEE International Conference on Robotics and Automation*, 751–756. Shanghai: Ieee.

Goda, K.; Kiyota, T.; Pokhrel, R. M.; Chiaro, G.; Katagiri, T.; Sharma, K.; and Wilkinson, S. 2015. The 2015 Gorkha Nepal Earthquake: Insights from Earthquake Damage Survey. *Frontiers in Built Environment* 1(April):1–15.

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

Goodchild, M. F., and Glennon, J. A. 2010. Crowdsourcing geographic information for disaster response: a research frontier. *International Journal of Digital Earth* 3(3):231–241.

Goodrich, M. A.; Cooper, J. L.; Adams, J. A.; Humphrey, C.; Zeeman, R.; and Buss, B. G. 2007. Using a Mini-UAV to Support Wilderness Search and Rescue: Practices for Human-Robot Teaming. In *IEEE International Workshop on Safety, Security and Rescue Robotics (SSRR)*, 1–6. Rome, Italy: IEEE.

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

He, R. 2007. *Planning in information space for a quadrotor helicopter in a GPS-denied environment*. Ph.D. Dissertation, MIT.

Hu, J.; Xie, L.; Xu, J.; and Xu, Z. 2014. Multi-agent cooperative target search. *Sensors (Switzerland)* 14(6):9408–9428.

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

Kolling, A., and Kleiner, A. 2013. Multi-UAV Motion Planning for Guaranteed Search. In *Autonomous Agents and Multiagent Systems*, 79–86.

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

Kothari, M.; Postlethwaite, I.; and Gu, D.-W. 2009. Multi-UAV path planning in obstacle rich environments using Rapidly-exploring Random Trees. In *Proceedings of the 48th IEEE Conference on Decision and Control (CDC) held jointly with 2009 28th Chinese Control Conference*, 3069–3074. Shanghai: IEEE.

Liu, Y. C., and Dai, Q. H. 2010. A survey of computer vision applied in aerial robotic vehicles. In *OPEE 2010 - 2010 International Conference on Optics, Photonics and Energy Engineering*, number 201, 277–280. Wuhan, China: IEEE.

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

Meier, P. 2015. Chapter 6: Uavs And Humanitarian Response. In *Drones And Aerial Observation: New Technologies For Property Rights, Human Rights, And Global Development: A Primer*, number July. www.newamerica.org, online edition. 103.

Morrow, N.; Mock, N.; Papendieck, A.; and Kocmich, N. 2011. Independent evaluation of the Ushahidi Haiti project. Technical report, Ushahidi.

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

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

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

Silver, D., and Veness, J. 2010. Monte-Carlo Planning in Large POMDPs. *Advances in Neural Information Processing Systems* 23:2164–2172.

United Nations Foundation. 2011. Disaster relief 2.0. Technical report, United Nations.

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