Mixed-Initiative Coordination for Disaster Response in the Real-World

Tracking No. 577

ABSTRACT

The problem of allocating emergency responders to rescue tasks is a key application area for agent-based coordination algorithms. However, to date, none of the proposed approaches take into account the uncertainty predominant in disaster scenarios and, crucially, none have been deployed in the real-world in order to understand how humans perform when instructed by an agent. Hence, in this paper, we propose a novel algorithm, using Multi-agent Markov Decision Processes to coordinate emergency responders. More importantly, we deploy this algorithm in a mixed-reality game within an agent that works alongside a human commander to guide human players to complete rescue tasks. In our field trials, our algorithm is shown to improve performance significantly and our results allow us to elucidate some of the key challenges faced when deploying of mixed-initiative team formation algorithms.

Keywords

Human-Agent Interaction, Agents Supporting Human Cooperation, Coordination, Decision under Uncertainty

1. INTRODUCTION

The coordination of teams of field responders in search and rescue missions is regarded as one of the grand challenges for multi-agent systems research [6]. In such settings, responders with different capabilities (e.g., fire extinguishing, digging, or life support) have to form teams in order to perform rescue tasks (e.g., extinguishing a fire or digging civilians out of rubble or both) to minimise costs (e.g., time or money) and maximise the number of lives and buildings saved. Thus, responders have to plan their paths to the tasks (as these may be distributed in space) and form specific teams to complete some tasks. These teams, in turn, may need to disband and reform other teams to complete other tasks requiring different capabilities, taking into account the status of the tasks (e.g., health of victims or building fire) and the environment (e.g., if a fire or radioactive cloud is spreading). Furthermore, uncertainty in the environment (e.g., wind direction or speed) or in the responders' abilities to complete tasks (e.g., some may be tired or get hurt) means that plans may need to change dynamically depending on the state of the players and the environment.

To address these challenges, in recent years, a number of algorithms and mechanisms have been developed to create teams and allocate tasks. For example, [9, 11] and [10, 4], developed centralised and decentralised optimisation algorithms respectively to allocate rescue tasks efficiently to teams of field responders with different capabilities. However, none of these approaches considered the uncertainty in the environment or in the field responders' abilities. Crucially, to date, while all of these algorithms have been shown to perform well in simulations (representing responders as computational entities), none of them have been *deployed* to guide *real* human responders (amateur or expert) in real-time rescue missions. Thus, it is still unclear whether these algorithms will cope with real-world uncertainties (e.g., communication breakdowns or change in wind direction), be acceptable to humans (i.e., agent-computed plans are not confusing and take into account human capabilities), and do help humans perform better than on their own.

Against this background, in this paper we develop a novel algorithm for team coordination under uncertainty and evaluate it within a real-world mixed-reality game that embodies the simulation of team coordination in disaster response settings. In more detail, we consider scenario involving rescue tasks distributed in a disaster space over which a radioactive cloud is spreading. Tasks need to be completed by the responders before the area is completely covered by the cloud (as responders will die from radiation exposure) which is spreading according to varying wind speed and direction. Our algorithm captures the uncertainty in the scenario (i.e., in terms of environment and player states) and is able to compute a policy to allocate responders to tasks to minimise the time to complete all tasks without them being exposed to significant radiation. The algorithm is then used by an agent to guide human responders based on their perceived states. This agent is then implemented in our deployed platform, AtomicOrchid, that structures the interaction between human responders, a human commander, and a planning agent in a mixed-reality location-based game. By so doing, we are able to study, both quantitatively and qualitatively, the performance of a human-agent collective (i.e., a mixed-initiative team where control can shift between humans and agents seamlessly) and the interactions between the different actors in the system. Thus, this paper advances the state of the art in the following ways:

1. We develop a novel approximate algorithm for team formation under uncertainty using a Multi-agent Markov Decision Process (MMDP) paradigm, and show how it accounts for real-world uncertainties.

2. We present AtomicOrchid, a novel platform to evaluate team formation under uncertainty using the concept of mixed-reality games. AtomicOrchid allows an agent, using our team formation algorithm, to coordinate, in real-time, human players using mobile phone-based messaging, to complete rescue tasks efficiently.

Appears in: *Proceedings of the 13th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2014), Lomuscio, Scerri, Bazzan, Huhns (eds.), May, 5–9, 2014, Paris, France.*

Copyright © 2014, International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). All rights reserved.

3. We provide the first real-world evaluation of a team formation agent in a disaster response setting in multiple field trials and present both quantitative and qualitative results. Our results allow us to elucidate some of the challenges for the formation of human-agent collectives.

When taken together, our results show, for the first time, how agent-based coordination algorithms for disaster response can be validated in the real-world. Moreover, these results allow us to derive a methodology and guidelines to evaluate human-agent interaction in real-world settings.

The rest of this paper is structured as follows. Section 2 formalises the disaster response problem as an MMDP. Section 3 then describes the algorithm to solve the path planning and task allocation problems presented by the MMDP while Section 4 describes the AtomicOrchid platform. Section 5 presents our pilot study and the evaluation of the system in a number of field trials. Finally, Section 6 concludes.

2. THE DISASTER SCENARIO

We consider a disaster scenario involving a satellite, powered radioactive fuel, that has crashed in a sub-urban area (see Section 4 to see how this helps implement a mixed-reality game). While debris is strewn around a large area, damaging buildings and causing accidents and injuring civilians, radioactive particles discharged in the air, from the debris, are gradually spreading over the area, threatening to contaminate food reserves and people. Hence, emergency services, voluntary organisations, and the military are deployed to help evacuate the casualties and resources before these are engulfed by a radioactive cloud. In what follows, we model this scenario formally and then describe the optimisation problem faced by the actors (i.e., including emergency services, volunteers, medics, and soldiers) in trying to save as many lives and resources as possible. We then propose an algorithm to solve this optimisation problem (in Section 3). In Section 4, we demonstrate how this algorithm can be used by a software agent (in our mixed-reality game) in a mixed-initiative process to coordinate field responders.

2.1 Formal Model

Let G denote a grid overlaid on top of the disaster space, and the satellite and actors are located at various coordinates $(x, y) \in G$ in this grid. The radioactive cloud induces a radioactivity level $l \in [0, 100]$ at every point it covers in the grid (100 corresponds to maximum radiation). While the exact radiation levels can be measured by responders on the ground (at every grid coordinate) using their geiger counter, it is assumed that some information is available from existing sensors in the area. However, this information is uncertain due to the poor positioning of the sensors and the variations in wind speed and direction (and we show how this uncertainty is captured in the next section). A number of safe zones $G' \subseteq G$ are defined where the responders can drop off assets and casualties. Let the set of n field responders be denoted as $p_1, \cdots p_i \cdots, p_n \in I$ and the set of ms rescue tasks as $t_1, \cdots, t_j, \cdots, t_m \in T$. As responders enact tasks, they may become tired, get injured, or receive radiation doses that may, at worst, be life threatening. Hence, we assign each responder a health level $h_i \in [0, 100]$ that decreases based on their radiation dose (h is decreased by $0.02 \times h_i$ per second) and assume that their decision to perform the task allocated to them is liable to some uncertainty (e.g., they may not want to do a task because they are tired or don't believe it is the right one to do). Moreover, each responder has a specific role $r \in Roles$ (e.g., fire brigade, soldier, or medic) and this will determine the capabilities he or she has and therefore the tasks he or she can perform. We denote as $Roles(p_i)$ the role of responder p_i. In turn, to complete a given task t_j, a set of responders $I' \subseteq I$ with specific roles $R_{t_j} \subseteq R$ is required. Thus, a task can only be completed by a team of responders I' if $\{Roles(p_i)|p_i \in I'\} = R_{t_j}$. In our scenario, we assume that the field responders are coordinated by the headquarters headed by a human coordinator H but assisted by a planner agent PA that can receive input from the human coordinator or the field responders. While the human coordinator H communicates its instructions directly to the responders (e.g., using an instant messaging client or walkie talkie), the planning agent PA can compute an allocation of tasks for the responders to complete. This is communicated to them in terms of simple "Do task X at position Y". The responders may not want to do some tasks (for reasons outlined above) and may therefore simply accept or reject the received instruction. These interactions are depicted in Figure 2.1.

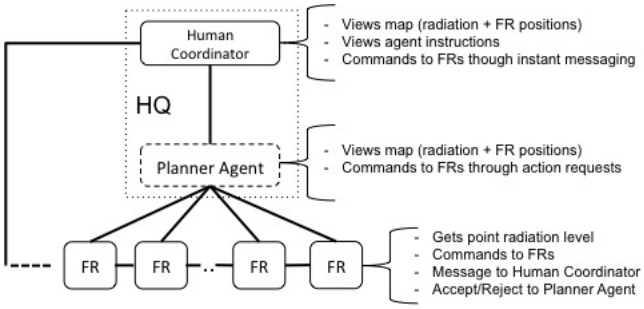

Figure 1: The interactions between different actors in the disaster scenario. Lines represent communication links. Planner agent and coordinator sit in the headquarters (HQ). Field responders (FRs) can communicate with all actors directly.

It is important to note that our model implements different types of control: (i) agent-based: when the agent instructs the responders (ii) human-based: when responders work with the coordinator or between themselves. Our model also captures different modes of control: (i) centralised: when responders respond to the planning agent or human coordinator (ii) decentralised: if responders coordinate between themselves. Crucially, this scenario allows for flexible levels of human and agent autonomy. For example, field responders may simply implement the plan given to them by the planner agent but can also feedback their constraints to the planner agent (as we demonstrate later) by rejecting some instructions and requesting new instructions.

Given this model, we next formulate the optimisation problem faced by the responders and solved by the planning agent (later in Section 3). To this end, we propose a Multi-Agent Markov Decision Process (MMDP) [3] that captures the uncertainties of the radioactive cloud and the responders' behaviours. Specifically, we model the spreading of the radiative cloud as a random process over the disaster space and allow the actions requested from the responders to fail (because they refuse to go to a task) or incur delays (because they are too slow) during the rescue process. This stands in contrast to previous work [10, 11] that require the process of task executions to be deterministic and explicitly model the task deadlines as deterministic constraints (which are stochastic in our domain). Thus in the MMDP model, we represent task executions as stochastic processes of state transitions. Thus, the uncertainties of the radioactive cloud and the responders' behaviours can be easily captured with transition probabilities. Additionally, modelling the problem as a MMDP allows us to use many sophisticated algorithms such as RTDP and MCTS that have already been well

developed in the literature [2, 7].

2.2 The Optimisation Problem

Here we formalise the optimisation problem that needs to be solved to coordinate the responders optimally. Hence, we define this problem as a Multi-agent Markov Decision Process (MMDP) [3] formally represented by tuple $\mathcal{M} = \langle I, S, \{A_i\}, P, R\rangle$, where I is the set of actors as defined in the previous section, S is the state space, A_i is a set of responder p_i's actions, P is the transition function, and R is the reward function. We elaborate on each of these below.

More specifically, $S = S_r \times S_{p_1} \times \cdots \times S_{p_n} \times S_{t_1} \times \cdots \times S_{t_m}$ where $S_r = \{l_{(x,y)} | (x, y) \in G\}$ is the state variable of the radioactive cloud to specify the radioactive level $l_{(x,y)} \in [0, 100]$ at every point $(x, y) \in G$. $S_{p_i} = \langle h_i, (x_i, y_i), t_j\rangle$ is the state variable for each responder p_i to specify his or her health level $h_i \in [0, 100]$, the coordinate (x_i, y_j), and the task t_j carried by the responder. $S_{t_j} = \langle st_j, (x_j, y_j)\rangle$ is then the state variable for task t_j to specify its status st_j (picked up, dropped off, or idle) and coordinate (x_j, y_j).

The three types of actions (in set A_i) a responder can take are: (i) *stay* in the current location (x_i, y_i), (ii) *move* to the 8 neighbouring locations, or (iii) *complete* a task located in (x_i, y_i). A joint action $\vec{a} = \langle a_1, \cdots, a_n\rangle$ is a set of actions where $a_i \in A_i$, one for each responder.

The transition function P is defined in more detail as: $P = P_r \times P_{p_1} \times P_{p_n} \times P_{t_1} \times P_{t_n}$ where:

- $P_r(s'_r|s_r)$ is the probability for the radioactive cloud to spread from state s_r to s'_r. It captures the uncertainty of the next radioactive levels of the environment due to the noisy sensor reading and the variation in wind speed and direction.
- $P_{p_i}(s'_{p_i}|s, a_i)$ is the probability for responder p_i to transit to a new state s'_{p_i} when executing action a_i. For example, when a responder is asked to go to a new location, he or she may not be there because he or she becomes tired, gets injured, or receives radiation doses that are life threatening.
- $P_{t_j}(s'_{t_j}|s, \vec{a})$ is the probability for task t_j. A task t_j can only be completed by a team of responders with required roles locating in the same coordinate as t_j.

Now, if a task t_j is completed (i.e., in s_{t_j}, the status st_j is marked as "dropped off" and the coordinate (x_j, y_j) is within a drop off zone), the team will be rewarded using function R. There will be a penalty for the team if a responder p_i gets injured or receives a high dose of radiation (i.e., in s_{p_i}, the health level h_i is 0). Moreover, we attribute a cost to each of the responders' since it will requires them to exert some effort (e.g., running or carrying objects).

Give the above definitions, a policy for the MMDP is a mapping from states to joint actions, $\pi : S \rightarrow \vec{A}$ so that the responders know which actions to take given the current state of the problem. The quality of a policy π is usually measured by its expected value V^π, which can be computed recursively by the Bellman equation:

$$V^\pi(s) = R(s, \pi(s)) + \gamma \sum_{s' \in S} P(s'|s, \pi(s))V^\pi(s') \qquad (1)$$

where $\pi(s)$ is a joint action given s and $\gamma \in (0, 1]$ is the discounted factor. The goal of solving the MMDP is to find an optimal policy π^* that maximises the expected value with the initial state s^0, $\pi^* = \arg\max_\pi V^\pi(s^0)$.

At each decision step, we assume the planning agent can fully observe the state of the environment s by collecting sensor readings of the radioactive cloud and GPS locations of the responders.

Algorithm 1: Team Coordination

Input: the MMDP model and the current state s.
Output: the best joint action $\vec{a}$.

```
// The task planning
{t_i} ← compute the best task for each responder i ∈ I
foreach i ∈ I do
    // The path planning
    a_i ← compute the best path to task t_i
return a⃗
```

Given a policy π of the MMDP, a joint action $\vec{a} = \pi(s)$ can be selected and broadcast to the responders (as mentioned earlier). By so doing, each responder can be instructed by the agent and know how to act in the field. In the next section we discuss the computational challenges of finding an optimal policy and propose a scalable approximation algorithm for this purpose.

3. TEAM COORDINATION ALGORITHM

The MMDP model proposed in the previous section results in a very large search space even for small-sized problems. For example, with 8 responders and 17 tasks in a 50×55 grid, the number of possible states is more than 2×10^{400}. Therefore, it is practically impossible to compute the optimal solution. In such cases, it is therefore better to consider approximate solution approaches that result in high quality allocations. The point of departure for our approximate solution comes from the observations that, when making a decision, the responders first need to *cooperatively* select a task to form a team with others (i.e., agree on who will do what), and they can *independently* compute the best path to the task. In our planning algorithm, we use this observation to decompose the decision-making process into a hierarchical structure with two levels:

- At the top level, a task allocation algorithm is run for the whole team to assign the best task to each responder given the current state of the world.
- At the bottom level, given a task, a path planning algorithm is run for each responder to find the best path to the task from his or her current location.

Furthermore, not all states are relevant to the problem (e.g., if a responder gets injured, he or she is incapable of doing any task in the future and therefore his or her states are irrelevant to other responders) and we only need to consider the reachable states given the current global state S of the problem. Hence, given the current state, we compute the policy online only for reachable states. This saves a lot of computation because the size of the reachable states is usually much smaller than the overall state space. For example, given the current location of a responder, the one-step reachable locations are the 8 neighbouring locations plus the current locations, which are 9 locations out of the 50×55 grid. Jointly, the reduction will be huge, which is 9^8 vs. $(50 \times 55)^8$ for 8 responders. Another advantage of online planning is that it allows us to tweak the model as more information is obtained or unexpected events happen. For example, if the wind speed increases or the direction of wind increases, the uncertainty about the radioactive cloud may increase. If a responder becomes tired, the outcome of his or her actions may be liable to more uncertainty.

The main process of our online hierarchical planning algorithm is outlined in Algorithm 1. The following sections will describe the procedures of each level in more detail.

3.1 Task Allocation

As described in Section 2.2, each responder p_i has a specific role $r_i \in Roles$ to determine which task he or she can perform and a task t can only be completed by a team of responders with the required roles $Roles(t)$. If, at some point in the execution of a plan, a responder p_i is incapable of performing a task (e.g., because she is tired or suffered a high radiation does), he or she will be removed from the set of responders under consideration that is $I = I \setminus p_i$ if pi. This information can be obtained from the state $s \in S$. When a task is completed by a chosen coalition, the task is simply removed from the set, that is $T = T \setminus t_k$ if t_k has been completed.

Now, in order to capture the efficiency of groupings of responders at performing tasks, we define the notion of a coalition C as a subset of responders, that is, $C \subseteq I$.[1] Thus, we can identify all possible coalitions $\{C_{jk}\}$ for each task t_j where $\{r_i | p_i \in C_{jk}\} = Roles(t_j)$. Crucially, we define the value of a coalition $v(C_{jk})$ that reflects the level of performance of a coalition C_k in performing task t_k. Then, the goal of task allocation algorithm is to assign a task to each coalition that maximises the overall team performance given the current state s, i.e., $\sum_{j=1}^{m} v(C_j)$ where C_j is a coalition for task t_j and $\{C_1, \cdots, C_m\}$ is a *partition* of I ($\forall j \neq j', C_j \bigcap C_{j'} = \emptyset$ and $\bigcup_{j=1}^{m} C_j = I$). In what follows, we first detail the procedure to compute the value of all coalitions that are valid in a given state and then proceed to detail the main algorithm to allocate tasks. Note that these algorithms take into account the uncertainties captured by the MMDP.

3.1.1 Coalition Value Calculation

The computation of values $v(C_{jk})$ for each coalition C_{jk} is challenging because not all tasks can be completed by one allocation (there are usually more tasks than responders) and the policy after completing task t_j must be computed as well and this is time-consuming. Here, we propose to estimate the value through several simulations. This is much cheaper computationally because we do not need to compute the complete policy in order to come up with a good estimate of the value of the coalition. According to the central limit theorem, as long as the number of simulations is sufficient large, the estimated value will converge to the true coalition value. The main process is outlined in Algorithm 2.

Algorithm 2: Coalition Value Calculation

Input: the current state s, a set of unfinished tasks T, and a set of free responders I.
Output: a task assignment for all responders.

1 $\{C_{jk}\} \leftarrow$ compute all possible coalitions of I for T
2 **foreach** $C_{jk} \in \{C_{jk}\}$ **do**
  `// The N trial simulations`
3 **for** $i = 1$ **to** N **do**
4 $(r, s') \leftarrow$ simulate the process with the starting state s until task k is completed by the responders in C_{jk}
5 **if** s' *is a terminal state* **then**
6 $v_i(C_{jk}) \leftarrow r$
7 **else**
8 $V(s') \leftarrow$ estimate the value of s' with MCTS
9 $v_i(C_{jk}) \leftarrow r + \gamma V(s')$
10 $v(C_{jk}) \leftarrow \frac{1}{N} \sum_{i=1}^{N} v_i(C_{jk})$
11 **return** the task assignment computed by Equation 3

In each simulation of Algorithm 2, we first assign the responders in C_{jk} to task t_j and run the simulator starting from the current state s (Line 4). After task t_j is completed, the simulator returns the sum of the rewards r and the new state s' (Line 4). If all the responders in C_{jk} are incapable of doing other tasks (e.g., having received too high a radioactive dose), the simulation is terminated (Line 5). Otherwise, we estimate the expected value of s' using Monte-Carlo Tree Search (MCTS) (Line 8), which provides good tradeoff between exploitation and exploration of the policy space and has been shown to be efficient for large MDPs [7]. After N simulations, the averaged value is returned as an approximation of the coalition value (Line 10).

The basic idea of MCTS is to maintain a search tree where each node is associated with a state s and each branch is a task assignment for all responders. To implement MCTS, the main step is to compute an assignment for the free responders (a responder is free when he or she is capable of doing tasks but not assigned to any task) at each node of the search tree. This can be computed by Equation 3 using the coalition values estimated by the UCB1 heuristic [1]:

$$v(C_{jk}) = \overline{v(C_{jk})} + c\sqrt{\frac{2N(s)}{N(s, C_{jk})}} \tag{2}$$

where $\overline{v(C_{jk})}$ is the averaged value of coalition C_{jk} at state s so far, c is a tradeoff constant, $N(s)$ is the visiting frequency of state s, and $N(s, C_{jk})$ is the frequency that coalition C_{jk} has been selected at state s. Intuitively, if a coalition C_{jk} has bigger averaged value $\overline{v(C_{jk})}$ or is rarely selected ($N(s, C_{jk})$ is smaller), it has higher chance to be selected in the next visit of the tree node.

As we assume that the role of a responder and the role requirements of each task is static, we can compute all possible coalition values offline and therefore, in the online phase, we only need to filter out the coalitions for completed tasks and those containing incapacitated responders to compute the coalition set $\{C_{jk}\}$.

3.1.2 Coalitional Task Allocation

Given the coalition values computed above, we then solve the following optimisation problem to find the best solution:

$$\begin{array}{ll} \max\limits_{x_{jk}} & \sum_{j,k} x_{jk} \cdot v(C_{jk}) \\ \text{s.t.} & x_{jk} \in \{0, 1\} \\ & \forall j, \sum_k x_{jk} \leq 1 \quad \text{(i)} \\ & \forall i, \sum_{j,k} \delta_i(C_{jk}) \leq 1 \quad \text{(ii)} \end{array} \tag{3}$$

where x_{jk} is the boolean variable to indicate whether coalition C_{jk} is selected for task t_j or not, $v(C_{jk})$ is the characteristic function for coalition C_{jk}, and $\delta_i(C_{jk}) = 1$ if responder $p_i \in C_{jk}$ and 0 otherwise. In the optimisation, Constraint (i) ensures that a task t_j is allocated at most to only one coalition (a task does not need more than one group of responders). Constraint (ii) ensures that a responder p_i is assigned to only one task (a responder cannot do more than one task at the same time). This is a standard MILP that can be efficiently solved using standard solvers such as IBM ILOG's CPLEX.

3.1.3 Adapting to Responders

One main advantage of our approach is that it can easily incorporate the preferences of the responders. For example, if a responder rejects a task allocated to it by the planning agent, we simply filter out the coalitions for the task that contain the responder. By so doing, the responder will not be assigned to the task. Moreover, if a responder prefers to do the tasks with another responder, we can increase the weights of the coalitions that contain them in Equation 3 (By default, all coalitions have identical weights of 1.0). Thus, our

[1]Here coalitions are not considered in the game-theoretic sense as all agents and coalitions aim to maximise the global objective.

approach is adaptive to various preferences of human responders. In particular, we show how the adaptive capability of our algorithm is used in AtomicOrchid in a real-world deployment (in Section 5) Next we show how the path of each responder is computed taking into account real-world uncertainties.

3.2 Path planning

In the path planning phase, we compute the best path for a responder to her assigned task. This phase is stochastic as there are uncertainties in the radioactive cloud and the responders' actions. We model this problem as a single-agent MDP that can be defined as a tuple, $\mathcal{M}_i = \langle S_i, A_i, P_i, R_i \rangle$, where: $S_i = S_r \times S_{p_i}$ is the state space. In this level, responder p_i only need to consider the states of the radioactive cloud S_r and his or her own states S_{p_i} in the MMDP. A_i is the set of p_i's actions. In this level, responder p_i only need to consider her moving actions. $P_i = P_r \times P_{p_i}$ is the transition function. In this level, responder p_i only need to consider the spreading of the radioactive cloud P_r and the changes of his or her locations and health levels when moving in the filed P_{p_i}, which are defined earlier in the MMDP. R_i is the reward function. At this level, responder p_i only needs to consider the cost of moving to a task and the penalty of receiving high radiation doses. This is a typical MDP that can be solved by many state-of-the-art MDP solvers. We choose the Real-Time Dynamic Programming (RTDP) [2] approach because it particularly fits our problem, that is, a goal-directed MDP with large number of states.

There are several techniques we use to speed up the convergency of RTDP. In our problem, the terrain of the field is static. Thus, we can initialize the value function $V(s)$ using the cost of the shortest path between the current location to the task location on the map, which can be computed offline without considering the radioactive cloud. This helps RTDP quickly navigate among the obstacles (e.g., buildings, water pools, blocked roads) without getting trapped in dead-ends during the search. Another speed up is also possible if, when traversing the reachable states, we only consider the responder's current location and the neighbouring points. This will further speed up the algorithm where the main bottleneck is the huge state space.

3.3 Simulation results

To test the performance of our algorithm, we compare it with a greedy and a myopic method for task planning using a simulator. For each method, we use our path planning algorithm to compute the path for each responder so that they can reach the task location as fast as possible. In the greedy method, the responders are uncoordinated and select the closest tasks that they can do. In the myopic method, the responders are coordinated to select the tasks but have no lookahead for the future tasks. Table 1 shows the results for a problem with 17 tasks and 8 responders on a 50×55 grid. As we can see, our MMDP algorithm can complete more tasks than the myopic and greedy methods. More importantly, our algorithm can guarantee the safety of the responders while in the myopic method, only 25% of the responders are survived and in the greedy method all responders are killed by the radioactive cloud. More extensive evaluations are beyond the scope of this paper as the focus is on the use of the algorithm in a real-world deployment to test how humans take up advice computed in sophisticated ways by an agent-based planner.

Table 1: Experimental results for the MMDP, myopic, greedy algorithms in simulation.

	MMDP	myopic	greedy
Completed Tasks	71%	65%	41%
Survived Responders	100%	25%	0%

4. THE ATOMIC ORCHID PLATFORM

In this section we describe Atomic Orchid platform used to embed the planning agent in order to trial mixed-initiative coordination in a real-world scenario. We adopt a serious mixed-reality games approach to counteract the limitations of computational simulations For example, Simonovic highlights that simulations may rely on unrealistic geographical topography, and most importantly, may not account for "human psychosocial characteristics and individual movement, and (...) learning ability" [12]. The impact of emotional and physical responses likely in a disaster, such as stress, fear, exertion or panic remains understudied in approaches that rely purely on computational simulation [5]. In contrast, our approach creates a realistic setting in which participants experience physical exertion and stress through bodily activity and time pressure, mirroring aspects of a real disaster setting [8]; thus providing confidence in the efficacy of behavioural observations regarding team coordination supported by a planning agent.

In more detail, Atomic Orchid is a location-based mobile game based on the fictitious scenario described in Section 2. Field responders are assigned a specific role (e.g. 'medic', 'transporter', 'soldier', 'ambulance'). In their mission to rescue all the targets from the disaster space, the field responders are supported by (at least) one person in a centrally located HQ room, and the planning agent that sends the next task (as described in the previous section) to the team of field responders. In what follows, we first present the player interfaces used, the interactions with the planning agent, and the modelling of the radiation cloud in the game.

4.1 Player interfaces

Field responders are equipped with a 'mobile responder tool' providing sensing and awareness capabilities in three tabs (geiger counter, map, messaging and tasks; see figure 4.1). The first tab shows a reading of radioactivity, player health level (based on exposure), and a GPS-enabled map of the game area to locate fellow responders, the targets to be rescued and the drop off zones for the targets. The second tab provides a broadcast messaging interface to communicate with fellow field responders and HQ. The third tab shows the team and task allocation dynamically provided by the agent that can be accepted or rejected. Notifications are used to alert both to new messages and task allocations.

Figure 2: Mobile field responder and HQ interfaces.

The HQ is manned by at least one player (the HQ comman-

der) who has at her disposal an 'HQ dashboard' that provides an overview of the game area, including real-time information of the players' locations (see figure 4.1). The dashboard provides a broadcast messaging widget, and a player status widget so that the responders' exposure and health levels can be monitored. The HQ commander can further monitor the current team and task allocations by the agent. Crucially, only the HQ has a view of the radioactive cloud, depicted as a heatmap where 'Hotter' (red) zones correspond to higher levels of radioactivity.

4.2 System architecture

AtomicOrchid is based on the open-sourced geo-fencing game MapAttack[2] that has been iteratively developed for a responsive, (relatively) scalable experience. The location-based game is underpinned by client-server architecture, that relies on real-time data streaming between client and server. Client-side requests for less dynamic content use HTTP. Frequent events, such as location updates and radiation exposure, are streamed to clients to avoid the overhead of HTTP. In this way, field responders are kept informed in near real-time. Finally, to build the mobile app, we adapted an existing MapAttack Android app.

4.3 Integrating the Planning agent

The planning agent as described in Section 3 takes the game status (i.e., positions of players, know status of the cloud, and messages received from players - see below) as input and produces a plan of optimal team-task allocation for the current state. The agent is deployed on a separate server. The AtomicOrchid server requests a plan from the agent via a stateless HTTP interface by transmitting the game status in JSON format. Polling (and thus re-planning) is triggered by two kinds of game events:

- *Completion of task*. On successful rescue of a target, a new plan (i.e., allocation of tasks to each responder) is requested from the agent.
- *Explicit reject*. On rejection of a task allocation by field responders, a new plan is requested. More importantly, the rejected allocation is, in turn, used as a constraint within the optimisation run by the planner agent (as described in Section 3.1.3).

Once a plan is pulled from the planning agent, the AtomicOrchid game engine splits the plan into individual task allocations for each player and send them to their mobile responder app. The app presents the task allocation in the task tab, detailing: 1) the responder to team up with, 2) the allocated target (using target id), and 3) approximate direction of the target (e.g., north, east).

Once a player accepts a task, an acknowledgement is sent to their teammates, while rejecting a task triggers a task assignment from the agent. Furthermore, the HQ commander is provided with a visualisation of task allocations for each player on demand (by button press), to help monitor the task allocation computed by the agent.

4.4 Radiation Cloud Modelling

The radiation cloud is assumed to be monitored using a number of sensors on the ground (within the disaster space) that collect readings of the radiation cloud intensity and wind velocity every minute of the game. These sensors can be at fixed locations or held by mobile agents. The radiation cloud diffusion process is modelled by a nonlinear Markov field stochastic differential equation,

$$\frac{D\text{Rad}(\mathbf{z},\tau)}{D\tau} = \kappa\nabla^2\text{Rad}(\mathbf{z},\tau) - \text{Rad}(\mathbf{z},\tau)\nabla\cdot\mathbf{w}(\mathbf{z},\tau) + \sigma(\mathbf{z},\tau)$$

[2]http://mapattack.org

where D is the material derivative, $\text{Rad}(\mathbf{z},\tau)$ is the radiation cloud intensity at location $\mathbf{z}$ at time τ, κ is a fixed diffusion coefficient and σ is the radiation source(s) emission rate. The diffusion equation is solved on a regular grid defined across the environment with grid coordinates $\mathbf{G}$. Furthermore, the grid is solved at discrete time instances τ. The cloud is driven by wind forces which vary both spatially and temporally. These forces induce anisotropy into the cloud diffusion process which is proportional to the wind velocity, $\mathbf{w}(\mathbf{z},\tau)$. The wind velocity is drawn from two independent Gaussian processes (GP), one GP for each Cartesian coordinate axis, $w_i(\mathbf{z},\tau)$, of $\mathbf{w}(\mathbf{z},\tau)$. The GP captures both the spatial distribution of the wind velocity and the dynamic process resulting from shifting wind patterns such as short term gusts and longer term variations. In our simulation, each spatial wind velocity component is modelled by a squared-exponential GP covariance function, K, with fixed input and output scales (although any covariance function can be substituted). Furthermore, as wind conditions may change over time we introduce a temporal correlation coefficient, ρ, to the covariance function. Thus, for a single component, w_i, of $\mathbf{w}$, defined over grid $\mathbf{G}$ at times τ and τ', the wind process covariance function is, $\text{Cov}(w_i(\mathbf{G},\tau), w_i(\mathbf{G},\tau')) = \rho(\tau,\tau')K(\mathbf{G},\mathbf{G})$. We note that, when $\rho = 1$ the wind velocities are time invariant (although spatially variant). Values of $\rho < 1$ model wind conditions that change over time.

Using the above model, we are able to create a realistic view of a radiation cloud moving over the disaster space, thus posing a real challenge both for the HQ (agent and commander) and the responders on the ground as predictions they can make of where the cloud will move to will be prone to uncertainty both to the simulated wind speed and direction.

5. REAL-WORLD EVALUATION

We ran three sessions of AtomicOrchid with participants recruited from the local university to trial mixed-initiative coordination in a disaster response scenario. The following sections describe the participants, procedure, session configuration and methods used to collect and analyse quantitative and qualitative data.

5.1 Participants and procedure

A total of 24 participants (7 of them were female) were recruited through posters and emails, and reimbursed with £15 for 1.5-2 hours of study. The majority were students of the local university. The procedure consisted of 30 minutes of game play, and about 1 hour of pre-game briefing, consent forms, and a short training session, and a post-game group discussion.

At the end of the briefing, in which mission objectives and rules were outlined, responder roles were randomly assigned to all participants (fire-fighter, medic, transporter, soldier). HQ was staffed by a different member of the research team in each session in order to mimic an experienced HQ whilst avoiding the same person running HQ every time. Moreover, responders were provided with a smartphone and HQ coordinators with a laptop. The team was given 5 minutes to discuss a common game strategy.

Field responders were then accompanied to the starting point within the designated game area, about 1 minute walk from headquarters. Once field responders were ready to start, HQ sent a 'game start' message. After 30 minutes of game play the field responders returned to the HQ where a group interview was conducted, before participants were debriefed and dismissed.

5.2 Game sessions

We ran one session without the planner agent, and two sessions with the planner agent to be able to compare team performance in

the two versions. We also ran a pilot study for each condition to fine tune game configuration. The 8 field responders in each session were randomly allocated a responder so that the whole team had two of each of the four kinds of responder roles. The terrain of the 400x400metres game area includes grassland, a lake, buildings, roads, and footpaths and lawns. There were two safe drop-off zones (i.e., radiation free) and 16 targets in each session. There were four targets for each of the four target types. The target locations, pattern of cloud movement and expansion were kept constant for all game sessions. The pilot study showed that this was a challenging, yet not too overwhelming configuration of game area size, and number of targets to collect in a 30 min game session.

5.3 Data collection and analysis

We developed a log file replay tool to triangulate video recordings of game action with the timestamped system logs that contain a complete record of the game play, including responders' GPS location, their health status and radioactive exposure, messages, cloud location, locations of target objects and task status.

In order to assess how humans interact with each other and with the agent, we focused on collecting data relevant to agent task allocations and remote messages that are used to support coordination. In particular, we use speech-act theory (Searle, 1975) to classify messages sent between and among responders and HQ. We focus on the most relevant types of acts in this paper (which are also the most frequently used in AtomicOrchid):

- Assertives: *speech acts that commit a speaker to the truth of the expressed proposition*; these were a common category as they include messages that contain situational information.
- Directives: *speech acts that are meant to cause the hearer to take a particular action*, e.g. requests, commands and advice, including task and team allocation messages.

We next detail the results of the trial and then discuss the implications of these results in Section 5.4.3.

5.4 Results

Overall, 8 targets were rescued in the non-agent condition (Session A), and respectively 12 targets (Session B), and 11 targets (Session C) were rescued in the agent condition. Teams (re-)formed six times in session A, four times in session B and nine times in session C. Player health after the game was significantly higher (more than twice better) for the agent-assisted sessions (80 for Session B and 82 for Session C) compared to the non-agent assisted session (40/100 in Session A). In fact, one responder 'died' in Session A. It was also found that the agent dynamically re-planned 14 times in session B, and 18 times in session C. Most of the times, this was triggered when a target was dropped off in the safe zone (24 times) – as this would free up resources for the algorithm to recompute an allocation – and sometimes this was triggered by a player rejecting the agent's task allocation (8 times) (as per Section 3).

Finally, Table 5.4 shows the directives (mainly related to task allocation and execution) and assertives (mainly related to situational awareness) sent in the sessions. The next sections draw on how these messages were handled to give a sense of mixed-initiative coordination in the game sessions.

5.4.1 Handling Task Allocations

Fig. 5.4.1 shows how field responders handled task allocations in the agent and in the non-agent condition. In the non-agent condition, HQ sent 43 task allocation directives. Out of these, the recipient field responders addressed only 15 messages (bringing them up

	no agent		agent				Total
Speech acts	Session A		Session B		Session C		
	HQ	FR	HQ	FR	HQ	FR	
Directives	89	0	34	2	34	0	159
Assertives	33	6	26	16	24	16	121
Total	122	6	60	18	58	16	280

Table 2: Message classification.

in conversation). Out of these 15, responders chose to ignore the instructions only once. The responder ignored the instruction because they were engaged in another task and did not want to abandon it. A further 4 HQ instructions were consistent with a decision to rescue a certain target that had already been rescued locally by the responders. In the remaining 10 cases, field responders chose to follow the instructions. Although players were willing to follow HQ's instructions, they failed to correctly follow the instructions due to confusion and misunderstanding in the communication. In fact, only 2 instances of directives from the HQ led to task completion.The field responders accomplished task allocation of the other 6 saved targets locally without being instructed by HQ.

In contrast, when task allocation was handled by the agent, responders accepted 24 tasks, out of which they completed 15 tasks successfully. Even if there was no response or consensus between the responders (in 17 cases), still six out of 17 tasks were completed successfully. In total, 20 task allocations were overridden by a the agent with a new task allocation.

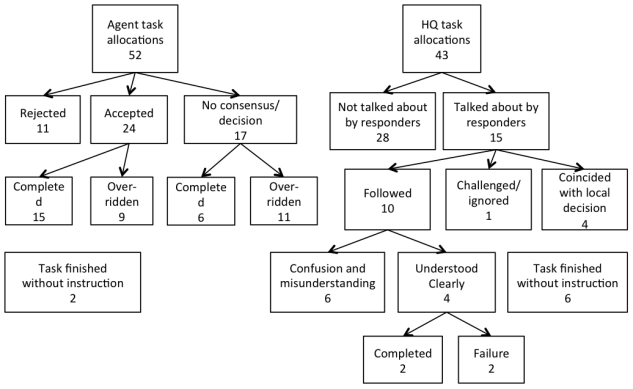

Figure 3: How task allocations were handled by field responders in the agent version (left) and in the non-agent version (right).

Rejecting task allocations.

Field responders rejected the agent's task allocation 11 times in the agent version. All of the rejections happened when the task allocation would have *split existing teams*, or instructed responders to team up with *physically more distant responders*. In most cases (9 out of 11), the rejection triggered re-planning and *adjusted the task allocation* to become consistent with the responder's current team. In the other 2 cases, rejected the task allocation one more time to receive the desired task allocation.

Overall, players were more likely to reject a plan if their proposed teammates were far away from them. For accepted instructions, the average distance between suggested teammates was 12 metres. For rejected instructions, the average distance between suggested teammates was 86 metres.

5.4.2 The Role of the HQ Commander

The burden of task allocation was borne by the HQ commander in the non-agent assisted setting. Instead, in the agent-assisted setting, the HQ commander frequently monitored the allocation of tasks returned by the agent (57 clicks on 'show task' in UI responder status widget). Whereas 43 directives in the non-agent session were task allocations, only 16 out of 68 directives were directly related to task allocations in the agent version. Out of these, HQ reinforced the agent instruction 6 times (e.g., "SS and LT retrieve 09"), and complemented the agent's task allocation 5 times ("DP and SS, as soon as you can head to 20 before the radiation cloud gets there first"). HQ did 'override' the agent instruction in 5 cases.

In the agent version, the majority of HQ's directives (52 out of 68) and assertives (49 out of 51) focussed on providing situational awareness and safely routing the responders to avoid exposing them to radiation. For example, "Nk and JL approach drop off 6 by navigating via 10 and 09.", or "Radiation cloud is at the east of the National College".

5.4.3 Summary and Guidelines

The results suggest three key observations with regard to mixed-initiative coordination in the trial:

- Field responders performed better (rescued more targets), and maintained higher health levels when supported by the agent. These results corroborate those obtained under simulation (see Section **??**) and showcase the superiority in the forward-planning capability of the planning agent compared to human responders.
- Rejecting tasks was a frequently employed method to to trigger re-planning to obtain new task allocations aligned with responder preferences. This showed that by engineering the planning agent to be adaptive to changes in the preferences or constraints of the responders, the planning agent was able to cope with the uncertainty predominant in the game.
- When task allocation was handled by the agent, the human HQ could focus on providing vital situational awareness to safely route field responders around danger zones; thus demonstrating division of labour and complementary collaboration between human and agent.

Given the above results we argue that a planning agent for team formation should not only model the uncertainty in player behaviours and the environment, but that interactional challenges need to be met in order to ensure proper acceptance of such a technology in practice. In particular, we propose the following design guidelines for human-agent collectives:

1. Adaptivity: our results suggest that planning algorithms should be designed to take in human input, and more importantly, be *responsive* to the needs of the users. As we saw in AtomicOrchid, players repeatedly requested new tasks and this would not have been possible unless the algorithm we had designed was computationally efficient but also had the ability to assimilate updates, requests, and constraints dynamically. We believe this makes the algorithm more acceptable.
2. Interaction Simplicity: our agent was designed to issue simple commands (Do X with Y) and respond to simple requests (OK or Reject Task). Such simple messages were shown to be far more effective at guiding players to do the right task than human communication that is fraught with inconsistencies and inaccuracies. In fact, we would suggest that agents should be designed with minimal options to simplify the reasoning users have to do to interact with the agent, particularly when they are under pressure to act.
3. Flexbile autonomy: the HQ dashboard proved to be a key tool for the HQ coordinator to *check* and *correct for* the allocations of the agent, taking into account the real-world constraints that the players on the ground faced. In particular, letting the human oversee the agent (i.e., "on-the-loop") at times and actively instructing the players (and bypassing the agent) at other times (i.e., "in-the-loop") as and when needed, was seen to be particularly effective. This was achieved by the HQ coordinator without the agent defining when such transfers of control should happen (as in [**?**]) and, therefore left the coordinator the option of taking control when she judged it was needed. Hence, we suggest that such deployed autonomous systems should be built for flexible autonomy, that is, interfaces should be designed to pass control *seamlessly* between human and agents.

6. CONCLUSIONS

In this paper we presented a novel algorithm for team formation under uncertainty and a study of the deployment of such an algorithm within a planner agent working alongside human responders in a new mixed-reality game platform called AtomicOrchid. The results in simulation show that our algorithm outperforms the baseline greedy algorithm significantly both in terms of number of tasks completed and number of responders unharmed by radiation. Moreover, we demonstrated in our real-world deployment the effectiveness of the planning agent at improving human coordination and how it can be made to collaboratively, with humans, come up with new plans given real-world constraints. Moreover, based on our results, we provided new guidelines for the design of human-agent collectives. Future work will look at exploring different interactional arrangements of humans and agents, in particular where control may be distributed across the team.

7. REFERENCES

[1] P. Auer, N. Cesa-Bianchi, and P. Fischer. Finite-time analysis of the multi-armed bandit problem. *Machine learning*, 47(2-3):235–256, 2002.

[2] A. G. Barto, S. J. Bradtke, and S. P. Singh. Learning to act using real-time dynamic programming. *Artificial Intelligence*, 72(1):81–138, 1995.

[3] C. Boutilier. Planning, learning and coordination in multi-agent decision processes. In *Proceedings of the 6th conference on Theoretical aspects of rationality and knowledge*, pages 195–210, 1996.

[4] A. Chapman, R. A. Micillo, R. Kota, and N. R. Jennings. Decentralised dynamic task allocation: A practical game-theoretic approach. In *Proc. of AAMAS 2009*, pages 915–922, May 2009.

[5] J. Drury, C. Cocking, and S. Reicher. Everyone for themselves? a comparative study of crowd solidarity among emergency survivors. *The British journal of social psychology*, 48(3):487–506, 2009.

[6] H. Kitano and S. Tadokoro. Robocup rescue: A grand challenge for multiagent and intelligent systems. *AI Magazine*, 22(1):39–52, 2001.

[7] L. Kocsis and C. Szepesvári. Bandit based monte-carlo planning. In *Proceedings of the 17th European Conference on Machine Learning*, pages 282–293, 2006.

[8] Pan American Health Organization. Stress management in disasters, 2001.

[9] S. D. Ramchurn, A. Farinelli, K. S. Macarthur, and N. R. Jennings. Decentralized coordination in robocup rescue. *Comput. J.*, 53(9):1447–1461, 2010.

[10] S. D. Ramchurn, M. Polukarov, A. Farinelli, N. C. Truong, and N. R. Jennings. Coalition formation with spatial and temporal constraints. In *AAMAS*, pages 1181–1188, 2010.

[11] P. Scerri, A. Farinelli, S. Okamoto, and M. Tambe. Allocating tasks in extreme teams. In *Proc. of AAMAS 2005*, pages 727–734. ACM, 2005.

[12] S. P. Simonovic. Systems approach to management of disasters: Methods and applications. In *Disaster Prevention and Management*, volume 20. Wiley, 2009.