HAC-ER: A Disaster Response System based on Human-Agent Collectives*

Sarvapali D. Ramchurn, Trung Dong Huynh, Yuki Ikuno, Jack Flann, Feng Wu, Luc Moreau, Nicholas R. Jennings
Electronics and Computer Science
University of Southampton
Southampton, UK

Joel E. Fischer, Wenchao Jiang, Tom Rodden
Mixed Reality Lab
University of Nottingham
Nottingham, UK

Edwin Simpson, Steven Reece, Stephen Roberts
Pattern Recognition Group
University of Oxford
Oxford, UK

ABSTRACT

This paper proposes a novel disaster management system called HAC-ER that addresses some of the challenges faced by emergency responders by enabling humans and agents, using state-of-the-art algorithms, to collaboratively plan and carry out tasks in teams referred to as human-agent collectives. In particular, HAC-ER utilises crowdsourcing combined with machine learning to extract situational awareness information from large streams of reports posted by members of the public and trusted organisations. We then show how this information can inform human-agent teams in coordinating multi-UAV deployments as well as task planning for responders on the ground. Finally, HAC-ER incorporates a tool for tracking and analysing the provenance of information shared across the entire system. In summary, this paper describes a prototype system, validated by real-world emergency responders, that combines several state-of-the-art techniques for integrating humans and agents, and illustrates, for the first time, how such an approach can enable more effective disaster response operations.

Categories and Subject Descriptors

I.2.11 [**Distributed Artificial Intelligence**]: Intelligent agents; H.5.2 [**User Interfaces**]: User-centered design

General Terms

Applications

Keywords

Innovative Applications; Human and Agents; Disaster Response.

1. INTRODUCTION

In the aftermath of major disasters (man-made or natural), such as the Haiti earthquake of 2010 or typhoon Haiyan in 2013, emergency response agencies face a number of key challenges [19]. First, it is vital to gain *situational awareness* of the unfolding event to determine where aid is required and how it can be delivered, given that infrastructure may be damaged. Useful information can come from a variety of sources, including people on the ground, relief agencies, or satellite imagery. However, making sense of this information is a painstaking process, particularly as the information sources are liable to noise, bias, and delays. Second, emergency response agencies typically need to gather additional information by deploying unmanned aerial vehicles (UAVs). Using multiple UAVs avoids risking human life but involves additional complexity in controlling the vehicles and visualising the information they feed back [2]. Tasks should be allocated to maximise the amount of information collected, whilst considering limited battery capacity and ensuring human coordinators are not overwhelmed by the need to manually operate individual UAVs. The third challenge is to use situational awareness to allocate relief tasks to emergency responders, for example, digging people out of rubble, moving water treatment units to populated areas, or extinguishing fires. It is crucial to consider the travelling time required for each task, as this blocks responders from performing other tasks [9]. However, the capabilities of individual responders must be considered to ensure that all tasks can be performed effectively and that no one is put in harm's way. For example, it may not be suitable to allocate medics to densely built-up areas where a fire is spreading, or to attend casualties during riots. Finally, given that the disaster environment is highly uncertain and liable to change significantly, it is crucial that emergency response agencies can track and verify the information and decisions that they use, allowing them to modify or reinforce the current course of action whenever new information is detected or previously trusted information is invalidated, e.g. through direct verification by other organisations.

Against this background, we propose a prototype disaster management system called Human-Agent Collectives for Emergency Response, or HAC-ER (pronounced 'hacker'), that demonstrates how humans and agents can be coalesced into teams called *Human-Agent Collectives* (HACs) [8] to address the above challenges. We designed our system collaboratively with emergency responders from Rescue Global[1] and other defence organisations in the UK, and trialled our system with over 100 users, to determine how HACs can support emergency response in different activities. In more detail, this paper first demonstrates a HAC that integrates crowdsourcing to gather, interpret and fuse information from both trusted agencies and members of the public on the ground, and thereby determine priority areas for responders. We then develop a system for multi-UAV coordination using a HAC, which involves both a distributed coordination algorithm and a number of human operators to prioritise search areas. Rescue targets identified by UAVs are then passed to a HAC composed of a planning agent and responders on the ground, who work together to determine a schedule for

*Watch a video on HAC-ER here: http://bit.ly/17aDRqt.

Appears in: *Proceedings of the 14th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2015), Bordini, Elkind, Weiss, Yolum (eds.), May 4–8, 2015, Istanbul, Turkey.*

[1] http://www.rescueglobal.org.

the completion of tasks. Finally, we employ a provenance tracking and analysis tool to allow the HACs to react to events and provide accountability for both human and agent-based decision making.

The rest of this paper is structured as follows. Section 2 discusses the decision making challenges addressed by HAC-ER. Section 3 presents our crowdsourcing support tool, while Section 4 describes our mixed-initiative UAV command interfaces. Section 5 then shows how to allocate emergency responders to rescue targets and Section 6 describes how information and decisions are tracked by a provenance manager to guarantee system reliability in dynamic environments. Finally, Section 7 concludes the paper.

2. DECISION MAKING IN DISASTER RESPONSE

Emergency response agencies are typically hierarchical, military-style organisations that employ the OODA framework[2] [5, 6]. A command and control structure is established whereby decision making is divided into strategic, tactical, and operational levels. The teams responsible for each are sometimes referred to as Gold, Silver, and Bronze respectively.[3] At the strategic (Gold) level, decision makers from all major response agencies involved decide on the main objectives of the response effort. At the tactical level, based on the specified objectives, the Silver command team decides on the allocation of resources and tasks to be carried out, while at the operational level, Bronze first responders (FRs), on the ground, determine the logistics required to carry out those tasks. Information gathered from the ground is also passed back up from Bronze, through Silver, to Gold.

In this paper, we focus on the challenges faced by the Silver and Bronze levels of these organisations respectively. The Silver commanders need to gain *situational awareness*, that is, gather information from the disaster scene, ensure this information is reliable, and then attempt to efficiently schedule resources and tasks on the ground to meet their objectives. Thus, situational awareness may be gathered using a combination of: (i) crowdsourced reports from members of the public, (ii) deployments of UAVs in collaboration with Bronze FRs to collect aerial imagery and locate key targets (iii) deployments of FRs to gather first-hand information, while carrying out response efforts in collaboration with other agencies and citizens.

These different information sources come with different levels of reliability and at different costs. For example, gathering data from online crowdsourcing platforms such as Twitter[4] or Ushahidi[5] only requires relatively cheap web technology, but the reports from such platforms can be posted by unverified sources. Instead, deploying UAVs or FRs on the ground helps gather more reliable data and information but this may turn out to be an expensive exercise if the UAVs get damaged, or even tragic if FRs are put in danger. Hence, it is important to avoid deploying FRs to the ground, and instead focus on gathering as much high quality data from crowds and UAVs as quickly as possible. To this end, we design HAC-ER with the aim to support the work of human emergency responders with a number of agent-based tools. In more detail, we first develop machine learning agents to annotate crowdsourced reports and generate heatmaps for Silver commanders to visualise key priority areas

[2]OODA stands for Observe-Orientate-Decide-Act. It is a well established information gathering and decision making process for deployments in dynamic environments.

[3]Variants of this organisational structure do exist but we find that the Gold, Silver, Bronze model is the most prevalent from our interactions with emergency responders such as Rescue Global and Hampshire County Council.

[4]www.twitter.com.

[5]www.ushahidi.com.

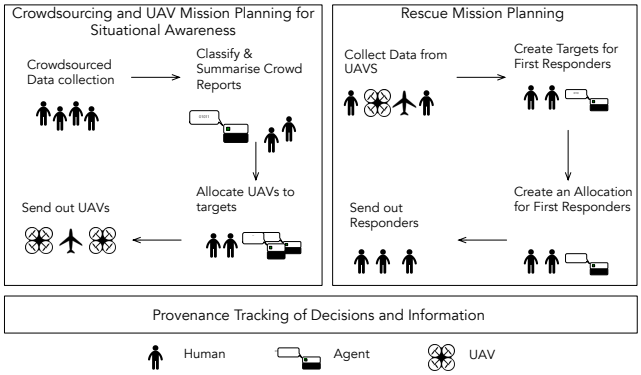


Figure 1: Information gathering and decision making process in the HAC-ER disaster management system.

(see Section 3). We then show how information from heatmaps can be used in determining UAV deployment plans that are generated using a decentralised coordination algorithm (see Section 4). Fig. 1 describes these two steps (top-left box) as part of an OODA loop, where the information gathered from the crowd (Observe) is used to decide on a plan for the UAV deployment (Orientate/Decide), which is then is carried out (Act).

During UAV missions, Silver operators at headquarters will typically monitor the video feeds coming back from the UAVs, while Bronze operators will supervise individual UAVs, and, at times, tele-operate them to gather more detailed information. As targets on the ground (e.g., casualties, collapsed buildings, fuel sources) are identified through this process, these targets are used by Silver commanders to allocate tasks to FRs. To help make these decisions, we developed interfaces for mixed-initiative task allocation, whereby human commanders interact with planning agents running coordination algorithms that exploit sensor data. Through this interaction, planning agents can compute plans that are efficient and *acceptable*, i.e. satisfy human preferences. The different steps of the mission planning process are graphically expressed in Fig. 1 (top-right box).

In general, a large amount of information is generated by various actors (humans, software agents, and UAVs) in a disaster response operation. Hence, a major contribution of this paper is the method by which provenance is tracked and used to improve the decision making process, providing accountability and ensuring dependencies between information and decisions are continuously recorded. This tracking system underlies all the decision making processes in our disaster management system (the bottom box in Fig. 1).

In this section, we have described how different components of HAC-ER fit into the organisation of emergency response agencies during major disasters. The following sections elaborate on each element of the HAC-ER system, namely the CrowdScanner, the Mixed-Initiative UAV Controller, the Mixed-Initiative Task Allocation System, and the Provenance Tracking System.

3. CROWDSCANNER

With the widespread popularity of mobile networks and the Internet, people affected by a disaster nowadays routinely post text messages or photographs to platforms such as Twitter or Ushahidi, reporting the situations on the ground in real time covering a wide area before FRs even arrive [12]. Thus, first-hand reports by members of the public has become a key source of information during a disaster, in addition to reports from FRs and aerial imagery of the effected areas. Vast quantities of data can be produced very rapidly as the disaster unfolds, which can overwhelm the silver commanders who require situational awareness to plan operations.

Only some of the information may be relevant and reports may be erroneous, out of date or duplicated (e.g. retweets). To overcome information overload, we design a software agent for situational awareness, the CrowdScanner, which uses machine learning to fuses heterogeneous reports into a common picture of the disaster, or a *heatmap of incidents*. Our approach is able to automatically combine information from both unreliable and trusted sources, filter out erroneous data, and help Silver commanders visualise the relevant information via a series of map overlays on a computer screen (see Fig. 2).

Our approach takes a large set of *unstructured data* from across a disaster zone, including geo-tagged text reports or images, and converts this to structured data with the help of a crowd of non-experts. The crowd answers key questions about each image or report and either (a) classifies it as one of several types of emergency, such as a collapsed building or medical emergency, or (b) indicates no emergency at that location, or (c) marks it as irrelevant. The crowd may also correct locations associated with reports if places are mentioned in the text do not correspond with their geo-tags. Machine learning algorithms then interpret this crowdsourced structured data to build a statistical model of the disaster area.

Our approach combines two key machine learning techniques, independent Bayesian classifier combination (IBCC) [17] and the Gaussian Process (GP) [14] into an algorithm that uses the principled Bayesian information framework to:

- efficiently combine classifications from different members of the crowd to remove erroneous information and rectify misclassifications,
- predict the location of emergencies across an entire disaster area by interpolating between sparse reports, and
- select subsets of reports to pass to the crowd for labelling and thus minimise the work undertaken by the crowd.

The IBCC algorithm combines crowd-classified reports from heterogeneous sources at each location to estimate the probability of an emergency at those locations. To do this, IBCC learns a *confusion matrix* [17] that encodes the reliability of each information source, such as an individual reporter or an NGO. By accounting for variations in accuracy of reports from different sources, we can fuse both highly trusted reports and weaker, error-prone data. Our prototype application demonstrates this by combining real Ushahidi reports written by people in Haiti after the 2010 earthquake, with simulated FR reports. Each FR is given a separate confusion matrix, but we do not have identifiers for the authors of Ushahidi reports. Instead, we have several classes of reports corresponding to different emergency categories,[6] which we treat as distinct information sources with separate confusion matrices. We can thereby account for the relevance of each type of report when predicting emergencies at a specific location.

IBCC does not require training with ground truth labels, but is instead an unsupervised approach that fits a model given only crowdsourced data and prior distributions over the confusion matrices and *emergency occurrence probability*, κ. We set the confusion matrix priors for Ushahidi reports to have a weak bias towards correctly indicating emergencies, encoding our initial uncertainty about their reliability. For the FRs, the priors are set to reflect our greater prior confidence in the accuracy of their reports. As more data is assimilated, IBCC is updated, and uncertainty in both κ and the confusion matrices decreases.

Now, we note that disasters can impact neighbouring areas in very similar ways. For instance, an earthquake will affect similar

[6]The Ushahidi dataset does not contain labels indicating 'no emergency here'; such reports were collected but not marked in the original Ushahidi project, although this would be useful in future.

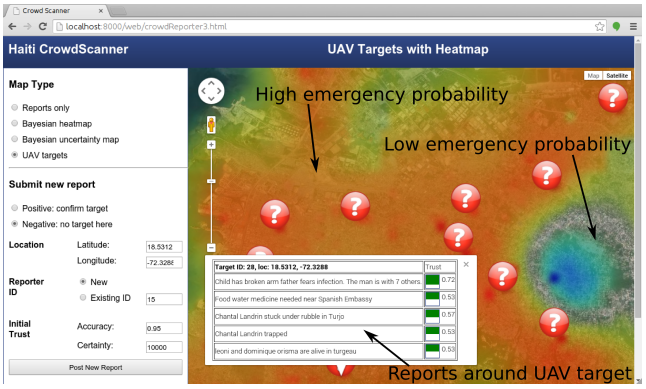


Figure 2: Heatmap user interface for Port-au-Prince after the 2010 Earthquake, showing high probability of emergency (red) and low probability (blue). The area marked as low emergency probability was identified using reports from FRs. Targets for UAVs are marked identified by red '?' icons.

infrastructure in neighbouring locations, and a flood can impact adjacent areas in similar ways. Hence, we extend the standard IBCC model to accommodate this insight into a novel algorithm that can make predictions at locations where we are missing reports. Our extended model assumes that the emergency occurrence probability varies reasonably smoothly from location to location. To model this, we assume that $\kappa(x, y) \in [0, 1]$ at coordinates (x, y) is drawn from a Gaussian process (GP), which is a distribution over smooth functions defined over the entire spatial disaster zone. The GP is mapped through a sigmoid function so that the distribution over $\kappa(x, y)$ is in the range $[0, 1]$ as detailed in [15]. We choose a low order isotropic, stationary Matérn covariance function to model the emergency occurrence probability over the two dimensional disaster zone as this does not impose a too stringent smoothness on $\kappa()$. The GP is fully integrated within the IBCC framework and consequently, inference of the combined IBCC/GP model is performed efficiently by variational Bayes. The length scales of the GPs are the most likely values found using the Nelder-Mead algorithm to optimise the variational lower bound. Our algorithm uses the GP to aggregate neighbouring reports and interpolate between them to determine the posterior distribution of $\kappa(x, y)$ across the entire space.

We plot the posterior distribution of emergencies over the entire disaster zone as a heatmap, as shown in Fig. 2, to highlight areas that require emergency aid. We can also show a heatmap of the variance in $\kappa()$, where high variance indicates regions for which little information about the emergency status is available. By fully exploiting information between neighbourhoods, our new method reduces uncertainty in the emergency situation at each location and consequently decreases the number of reports that must be labelled by the crowd. Furthermore, we use our algorithm to prioritise crowd labelling of reports at highly uncertain locations, and, when such reports are not available, to identify locations for further reconnaissance. From the heatmaps, we can automatically extract targets for UAVs to gather aerial imagery from likely emergency locations, to either confirm the precise nature of the emergency or invalidate reports and refocus response efforts. Targets are extracted by marking the peaks of the heatmap within the flight range of a UAV. These are depicted by red '?' icons in Fig. 2. The next section details our HAC approach for deploying UAVs to these targets.

4. MIXED-INITIATIVE UAV CONTROLLER

The targets suggested by the CrowdScanner come with varying degrees of certainty. Hence, the emergency response team will aim to verify these targets with first-hand knowledge. UAVs are typically used for this purpose to avoid putting personnel in harm's way. However, in most cases, the number of UAV operators available will be limited and the team will aim to send as many UAVs out as possible to gather information as quickly and effectively as possible. Hence, in what follows, we describe HAC-ER's UAV mission planning and command system that provides Silver commanders with supervisory control to allocate multiple UAVs to fly over points of interest in a disaster area so as to verify the potential targets. Moreover, we develop interfaces for low-level UAV teleoperation by individual Bronze operators on the ground to identify specific items of interest from the UAVs' camera feeds.

In more detail, the interaction between Silver and Bronze operators is mediated through voice-based communication as well as interface elements (see Section 4.2). This is an important part of the system as Bronze and Silver operators may have diverging views on how to operate the UAVs. For example, in the dynamic conditions of a disaster scenario, a Bronze operator may ground a UAV if she thinks the weather conditions are inappropriate, or she may take control to focus the UAV on a particular area to gather imagery. This may disrupt the plan decided by the Silver operators. We address this in Section 4.2.3 by designing the interaction to support dynamic handover of control between Silver and Bronze.

Furthermore, the human team is supported by coordinating agents that are individually in charge of the UAVs. More specifically, agents employ a decentralised coordination algorithm to allocate tasks among themselves. Thus, Silver operators are able to specify goals for the UAVs to achieve (fly to a point, scan a region) and the algorithm (distributively run by individual agents) allows the UAVs to decide which of them is best suited for each task. Crucially, our approach coordination implements the notion of flexible autonomy [8], whereby the agents' plan can be influenced by the human operators. We elaborate on this in the following section.

4.1 Flexible Decentralised Coordination

The flexible coordination module continuously monitors the state of the UAVs and tasks defined in the system and dynamically determines a task allocation plan to minimise the time that the UAVs take to complete their allocated task(s). We employ *max-sum* as the de facto coordination algorithm given that UAVs are naturally distributed in the scenario. As shown in [16], max-sum provides good approximate solutions to challenging dynamic decentralised optimisation.[7] However, max-sum does not explicitly handle constraints imposed by human operators. For example, if after running max-sum, agent A is tasked to go to point X, agent B to point Y, and agent C to point Z, there is no explicit method for human operators to partially modify the plan such that agent A goes to point Y, and B and C *automatically* re-allocate points Y and Z among themselves in the best way possible. Hence, to cater for such situations, we extend the max-sum algorithm to include constraints specified by human operators. In what follows, we provide a brief overview of the max-sum algorithm.[8]

4.1.1 The Max-Sum Algorithm

The max-sum algorithm works by first constructing a factor graph representation of a set of tasks (each representing a point or waypoints UAVs are meant to fly to) and the set of agents (each representing a UAV) and then sets a protocol for an exchange of messages between different nodes in the factor-graph. The factor graph is a bipartite graph where vertices represent agents and tasks, and edges the dependencies between them. Given a set of tasks, D, max-sum determines a subset of these tasks $D_i \subseteq D$ that are most appropriate for each UAV, i, using branch-and-bound techniques [16]. Effectively, this means pruning the factor graph to generate an acyclic graph over which max-sum is guaranteed to converge to a solution. Given this graph, agent and task nodes exchange messages that capture the utility of different allocations. Eventually, each agent node determines its best allocation by maximising over the sum of all messages it receives.

4.1.2 Integrating Human Input

Using a utility function defined from the time required for a UAV to fly to tasks, the priority of each task, and its urgency, max-sum allocates each UAV to a task to maximise the overall utility as per [3]. However, this assignment may not be accepted by the human operators as it may not consider the qualitative and quatitative priorities that humans attribute to tasks [18] as well as flight paths. For example, a UAV may be allocated by max-sum to fly from its position in the East to a task in the West but the human operators may, instead, prefer a UAV to fly from the South to the same task to provide imagery over the area covered by that path, which may be more important than the lateral traversal from East to West.

Against this background, given a plan computed by max-sum, through our planner interfaces (see Section 4.2), users can specify *manual* allocations of UAVs to tasks. These manual allocations specify a task-agent pair. Given this, for each agent i, we then define a set of tasks $D_i = \{j\}$. This effectively results in the deletion of all edges in the factor graph that connect the agent node i with other task nodes apart from that of j. This, in turn, forces max-sum to only allocate agent i to task j, and if two or more agents are required by task j, another agent will be chosen based on this restriction.

4.2 Interaction Design

We designed a number of fully functional, web-based user interfaces to allow a human-agent team to coordinate the UAVs. Through these interfaces, Silver operators visualise the plans suggested by max-sum and modify these plans as required. We also provide an interface for the Bronze operator to tele-operate a selected UAV. We next detail the interactions within each view.

4.2.1 Camera View

The camera view provides multiple live video streams from the UAVs (an MPEG stream is available from typical UAV camera modules). In a simulation of the system, we employed Google Maps[9] *aerial* view (see left of Fig. 3 for feeds from six UAVs). The images displayed are taken at real GPS locations of the UAVs in the disaster area. Targets, as identified by the CrowdScanner, are positioned at specific points in the area considered and displayed on the aerial view whenever the UAV flies over it. The user can then click on the map and create an annotation with the matching description as perceived by the Silver or Bronze operator. Once a target has been identified, an icon describing the target is then displayed across all views to ensure immediate situational awareness across the team.

An important feature of the interface is the flagging system, which Silver operators can use to alert Bronze operators when specific items of interest appear in the camera view. A special (clickable) button on each camera view highlights to the Bronze operator that

[7] Other decentralised coordination algorithms could be used here (e.g., DPOP, ADOPT, BnB-ADOPT) as we only adapt the tree over which they run to compute a solution.

[8] A detailed description of max-sum is beyond the scope of this paper. The reader is referred to [3, 10, 16] for more details on the implementation of max-sum for UAVs and task allocation domains.

[9] `http://maps.google.com`.

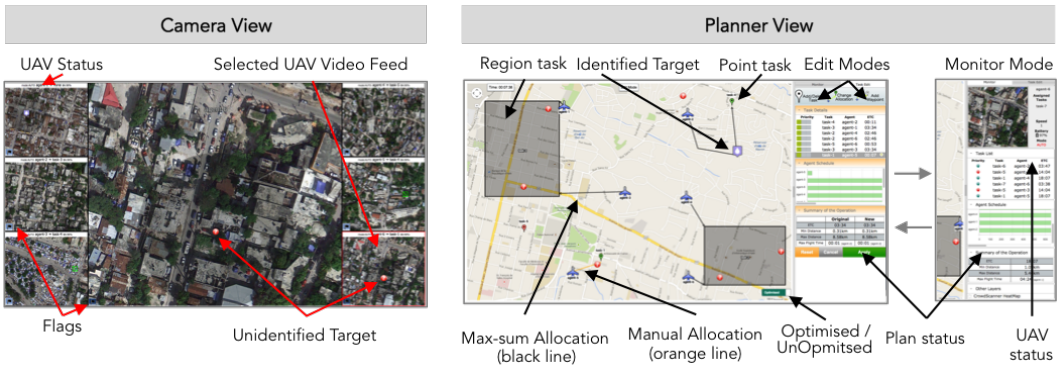


Figure 3: Silver operators' views including the camera view and the two modes of the planner view.

a specific UAV needs attention (see Section 4.2.3). Moreover, if a Bronze operator takes over control of a particular UAV, this UAV's camera view updates its status to 'tele-operated'. By so doing, we allow Silver and Bronze operators to coordinate their actions.

4.2.2 *Planner View*

This is the main planning tool that provides both monitoring and planning capabilities (see right of Fig. 3). Operators can choose to create two types of tasks for the UAVs; *point* tasks with way points for UAVs to fly to specific points in the space and scan the area along the way, and *region* tasks that define areas that teams of UAVs can self-organise to sweep-scan (i.e., they automatically divide the area between themselves and scan their individual sections). The operator can decide, either using max-sum or manually, which UAV should go to each of these tasks. These capabilities are accessible in two modes accessible through the tabs on the top right, namely 'Monitor' and 'Task Edit'. We describe each of these modes in turn.

Monitor Mode

This mode shows the current status of the allocation (see the right part of Fig. 3). The allocation of UAVs to tasks is represented as lines with arrowheads. Region tasks are marked as grey boxes and point tasks using icons. Paths chosen by the max-sum algorithm are shown in black, while paths chosen manually by the users are shown in orange. Once a region task has been completed, the grey box turns green. A region task is deemed completed when UAVs have covered its area, and a point task is considered completed when the allocated UAV has reached that task and hovered for 5 seconds. Once a point task is completed, the task disappears from the map. The right side of the monitor displays the current allocation of agents to tasks, the expected completion times and the schedule (as a Gantt chart) of the UAVs going to the tasks.

Task Edit Mode

This mode provides the user with a number of planning options (see right part of Fig. 3) through a number of sub-modes. The user can:

1. add/delete tasks (region or point): Users can create two types of tasks: (i) region tasks — this task requires two UAVs to carry out a sweep scan of the area selected by the user, and (ii) point tasks — a point selected in the map.
2. change/adapt the allocation of tasks to agents: the allocation automatically computed by max-sum can be changed by the user by clicking on a UAV and allocating it to another task. Max-sum then adapts its allocation to fit to the constraint set by the user (as per Section 4.1.1). For each allocation, a straight line is drawn from the selected UAV to the selected task (unless way points are specified).
3. add way points to the paths taken by the UAVs: this applies to paths chosen to point tasks, whereby users can adjust the path taken by a UAV to cover areas in more complex ways than in region tasks.

Once an allocation of UAVs to tasks has been chosen, the user can verify the completion time of the tasks using the side bar widgets and then decide to execute the plan.

4.2.3 *Bronze Operator View*

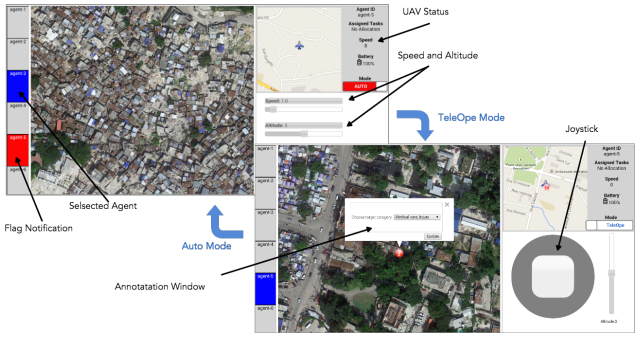


Figure 4: Tablet-based Bronze Operator Views.

The Bronze operator view is displayed on a tablet interface. In this view (see Fig. 4), the Bronze operator can select the specific UAV she may want to tele-operate or supervise more closely. Each UAV's camera view is accessible under different tabs (left of the screen). Additionally, in this view, we provide a notification mechanism for the Silver commanders to alert the Bronze operators, whereby the tab related to a specific UAV can be made to flash when the Silver commander 'flags' the UAV in their screen.

The view also incorporates a simulated joystick that controls the direction and speed of the UAV (pushing further in a given direction speeds up the UAV) and a slider that regulates the altitude of the UAV. The Bronze view is designed to receive live video feed from any drone that transmits an MPEG video stream.

5. MIXED-INITIATIVE TASK ALLOCATION

Having confirmed the locations of key targets in the disaster area through the UAVs, we next consider the deployment of FRs on the ground. More specifically, in this section we describe the system for Silver commanders to compute task allocations for Bronze FRs using the help of a planning agent. In this section, we first provide an overview of the model used by the planner agent, then describe the interaction mechanisms for the planner agent to allocate tasks to the FRs.

5.1 The Planner Agent

We developed an algorithm for a planner agent, that computationally models the behaviour of FRs in terms of the actions they take and the teams they form to complete their tasks. In contrast to the UAV task allocation problem where operators (Bronze or Silver) control UAVs at will, allocating human FRs requires judging whether they are fit to perform their tasks and whether there are any constraints that prevent them, individually, from doing so.

Given such uncertainties due to human behaviour, we model the task allocation problem using decision theoretic techniques. In more detail, the algorithm receives GPS locations of targets from the Mixed-Initiative UAV Controller and the location of FRs through their mobile responder tool (see Section 5.2.2). We model the problem of allocating FRs to targets using a *Multi-Agent Markov Decision Process* (MMDP). In what follows we only describe the MMDP model we use to solve the planning problem as the implementation details are beyond the scope of this paper.

Formally, an MMDP is defined as tuple $\langle I, S, \{A_i\}, T, R, s^0, \gamma \rangle$, where: $I = \{1, 2, \cdots, n\}$ is the set of n FRs as described above; S is a set of system states (e.g., where the FRs are positioned, their current task); A_i is the action set of FR $i \in I$; $T : S \times \vec{A} \times S \rightarrow [0, 1]$ is the transition function where $\vec{A} = \times_{i \in I} A_i$ is the set of joint actions; $R : S \times \vec{A} \rightarrow \Re$ is the reward function (e.g., the level of completion of a rescue mission or the time it takes to distribute vital resources); $s^0 \in S$ is the initial state; and $\gamma \in (0, 1]$ is the discount factor. Here, an action $a_i \in A_i$ is what an FR can do in one step in a fixed amount of time so all FRs complete their actions at the same time as commonly assumed in other MMDP applications. If some task takes much longer than others, FRs only need to repeat their actions several times until the task is finished. The outcome of solving an MMDP is a policy $\pi : S \rightarrow \vec{A}$ that maps from states to joint actions. Starting in state s^0, a joint action $\vec{a}$ is selected based on policy π. Each agent executes its component a_i of the joint action and the system transitions to next state s' based on the transition function. This process repeats with the new state s'. The objective of solving an MMDP is to find a policy that maximises the discounted expected values.

This MMDP can be fed to standard solvers (e.g., UCT [1]). However, this will be very inefficient due to the large search space of the model. Hence, we decompose the decision-making process into a hierarchical planning process: at the top level, a task planning algorithm is run for the whole team to assign the best task to each FR given the current state of the world; at the lower level, given a task, a path planning algorithm is run by each FR to find the best path to the task from her current location. Furthermore, since not all states of MMDPs are relevant to the problem, we only need to consider the reachable states given the current state. Hence, we compute the policy online, starting from the current state. This reduces computation significantly because the number of the reachable states is usually much smaller than the overall state space.

In more detail, we define the team values that reflect the level of performance of FR teams in performing tasks. This is computed from the estimated rewards that the teams obtain for performing the tasks. The expected values after completing the tasks are estimated by Monte-Carlo simulations. Given the team values, we assign a task to each team by solving a *mixed integer linear program* that maximises the overall team performance given the current state, subject to the requirements of each task to FRs. In the path planning phase, we compute the best path for a FR to her assigned task. Since there are uncertainties in the environment and the responders' actions, we model this problem as a single-agent MDP that can be solved by *real-time dynamic programming*. By so doing, we assign tasks to FRs such that their long term effects are rewarding while reduce the search space to a tractable size.

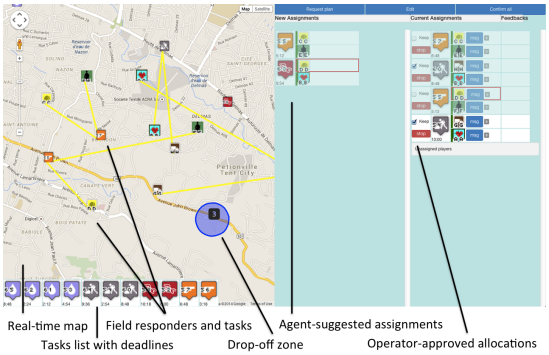


Figure 5: The Silver commanders' task allocation interface.

The output of our algorithm is a set of actions that describes which task each FR should undertake and their best paths given the tasks in the current state. This plan can then be provided on demand to Silver commanders. The challenge, however, is that such plans do not factor (i) whether the FRs are tired — if FRs are tired they may not be able to do the task allocated and may prefer to do easier (closer) tasks, and (ii) existing relationships between FRs — this can result in some FRs preferring to work together and leave others out. Crucially, it is not possible for the planner agent to model all the aspects of human collaboration and perception which could mean that plans may not make sense in the real world. Hence, in the next section we develop methods for interactions with the planner agent as well as between Silver commanders and Bronze FRs to help them converge on an effective plan.

5.2 Interaction Design

Following the Haiti scenario, once the UAVs have identified the targets on the ground (see Section 4), FRs with specific roles have to be allocated to these targets to further investigate, evacuate, rescue, repair, etc., depending on the nature of the target. To exemplify how this system would work, we assume four different types of FR roles (medics, fire fighters, soldiers, and transporters), and four different types of targets (injured personnel, social unrest, infrastructural damage, and water shortage). Mimicking real-world complexity, different targets have different role requirements, e.g., to rescue an 'injured personnel' target, a medic and a transporter are required. For Silver commanders, this creates the aforementioned task allocation problem under time pressure (due to task deadlines). In order to support Silver commanders with their coordination work to allocate tasks to FRs, we designed a set of interactive tools that both integrate the agent-based task allocation and path planning algorithm in the previous section, as well as situation awareness and communication capabilities for Silver commanders and FRs.

Fundamentally, these capabilities are enabled by two applications: (a) a web-based task allocation interface to support Silver coordination of FRs; and (b) a Mobile Responder Tool for FRs to respond to task assignments and messages from Silver commanders, to find each other, and to navigate the environment to evacuate targets. The following sections illustrate the ways in which we implemented a *human-in-the-loop* rationale for intelligent task allocation. Findings from earlier user studies showed that an effective interaction strategy leaves routine task assignment to the agent, but human input is required to confirm allocations or change these on demand, and, in particular, to deal with contingencies that may arise in the disaster setting.

5.2.1 Task allocation UI

The web-based task allocation interface for Silver commanders is depicted in Fig. 5. The operator uses the real-time map to locate responders and targets 'on the ground', to keep track of tasks and

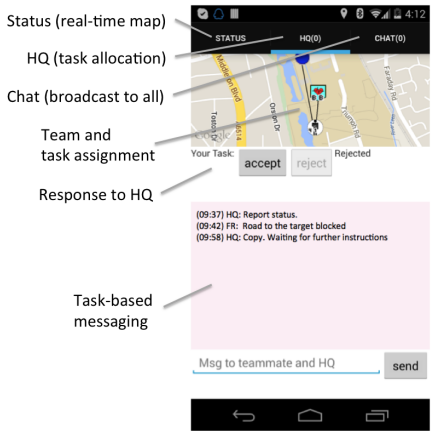


Figure 6: The Mobile Responder Tool.

their deadlines, to request assignments from the agent, and to inspect and confirm which of them get sent to the FRs. The task assignments may be edited 'manually', for example to prioritise a specific target due to type or deadline.

Due to relative complexity of the 'work flow' and the resulting target states, we implemented interactive elements to make the UI effective and guide the attention of the operator as follows:

- **FR feedback**. FRs accept or reject tasks (e.g., due to local knowledge unavailable to the operator). Rejection leads to a red, flashing signal, signalling to the operator that immediate attention is required. The operator may then decide to call upon the planner agent for a new allocation based on certain constraints or manually create an assignment of tasks to FR.
- **Hover-over alignment**. Hovering the cursor over task or FR icons in the task list or in the allocation list highlights the corresponding icon on the map, so as to facilitate aligning the views.
- **Drag-and-drop editing**. Silver commanders allocate FRs to tasks by drag-and-drop. Once a task is dropped into the assignment column, the responders' required roles are visualised to further guide the operator.
- **Task-based comms channels**. Operators have a channel for each team allocated to a task, to provide task-specific messaging (see Fig. 6).

5.2.2 Mobile Responder Tool

The mobile responder tool is depicted in Fig. 6. It provides the Bronze FRs with the same real-time map as the Silver commanders, with some convenience methods to facilitate focusing important elements on the limited mobile screen size, such as 'find me' and 'show task'. In addition, it provides task allocation information in a separate tab (shown), which users have to accept. In case they reject the allocation, they are first alerted to the task deadline in a modal ('are you sure?' dialogue) so as to discourage rejection.

Our pilot studies of different versions of our Mixed-Initiative Task Allocation system have demonstrated that FRs are more likely to accept plans computed by the planner agent if the plans are first analysed, modified, and validated by the Silver commanders [13]. Moreover, in our workshops with emergency responders from Rescue Global, it was particularly highlighted that such a disaster response system is an opportunity to provide, in real-time, an up-to-date picture of the disaster zone. Crucially, they highlighted the fact that the information generated by FRs on the ground, UAVs, and the CrowdScanner, needed to be tracked and analysed continually to identify potential discrepancies in decision making. Given

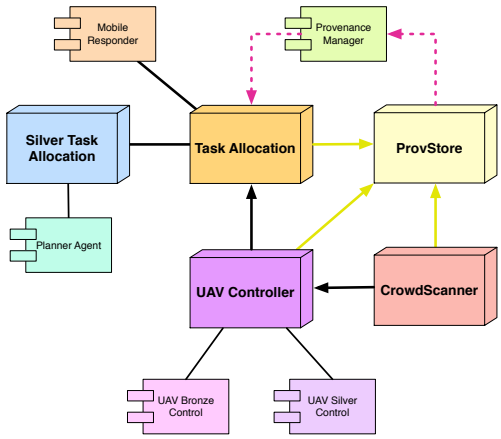


Figure 7: Information flows between HAC-ER's components.

this, we discuss next our approach to managing such information and decisions within the HAC-ER system.

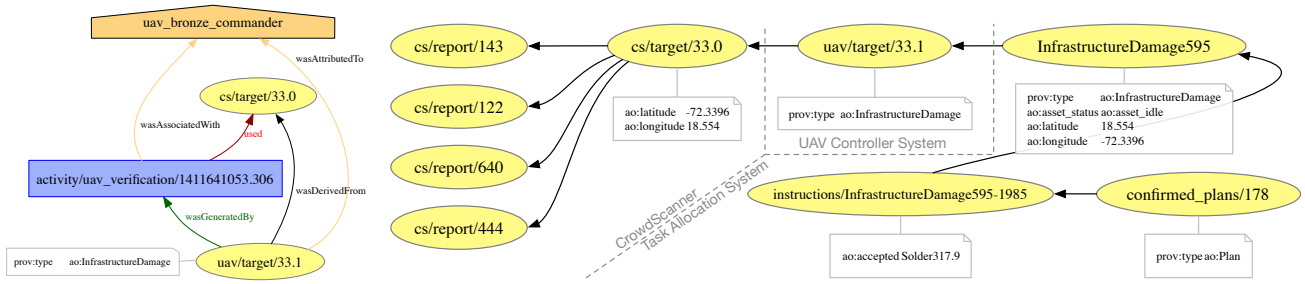
6. TRACKING INFORMATION AND DECISIONS IN HAC-ER

As discussed earlier, HAC-ER consists of loosely-coupled components that involve collectives of humans and agents. For an overview, Fig. 7 presents the components and information flows between them. Given the significant costs of making mistakes in our domain, tracking provenance of the information fed into decision making is a critical requirement. In this section, we describe how provenance in HAC-ER is tracked (Section 6.1) and used to improve awareness of changes across its components (Section 6.2).

6.1 Tracking Provenance

The World Wide Web Consortium (W3C) defines provenance as "a record that describes the people, institutions, entities, and activities involved in producing, influencing, or delivering a piece of data or a thing" [11]. In this system, when a piece of information is produced by one of the components, it records which inputs were used in the production of that piece of information and the agent(s) and/or human(s) that were involved. Fig. 8a shows an example of such provenance. In the example, the entity `uav/target/33.1` was generated by a "UAV Verification" activity; it was attributed to the UAV Bronze commander, was derived from another entity called `cs/target/33.0` (which was previously created by the CrowdScanner, but this is not shown in the figure), and has a property to represent its type as an "Infrastructure Damage." Examining this provenance, either when the entity `uav/target/33.1` is used or in a much later audit when the operation has finished, allows us and to track back to the origin of the information and to answer questions such as "who was responsible for the information", "on which other information it depended."

In our system, the provenance of information is stored in a purpose built repository for provenance, called ProvStore [7]. Individual components (i.e., the CrowdScanner, UAV Controller, and Task Allocation) record the provenance of information and data generated in each of their activities and report the provenance to ProvStore once the activity completes. The provenance of any entity can then be retrieved from ProvStore when required. Fig. 8b, for instance, shows the result of a query for all the dependencies of the entity `comfimed_plan/178`, which is a task allocation plan that has been confirmed by a commander. As can be seen, the result goes back all the way to the crowd reports aggregated by the CrowdScanner (some entities omitted due to space constraints).



(a) The provenance of a target (`uav/target/33.1`).

(b) A chain of derived information tracked across HAC-ER's three components.

Figure 8: Provenance graphs in HAC-ER. In (a) ellipses are entities, boxes are activities, and houses are agents.

Should any information in a derivation chain as the one shown in Fig. 8b later be discovered unreliable or incorrect, it would be possible to assess its potential impact by querying from ProvStore all of its dependents. This is indeed what our Provenance Manager monitors in order to ensure that the responders on the ground and their commanders are aware of potentially adverse information in dynamic situations typically in the area of disaster response. In the next section, we describe how this is achieved.

6.2 Monitoring Decisions

In an ongoing operation, the available information is typically uncertain and/or incomplete [19]. In HAC-ER, for instance, targets identified from aggregating crowd reports are inherently uncertain (see Section 3); new (and more trustworthy) reports from FRs, for example, can invalidate targets that are already assigned to UAVs. UAV operators can also make mistakes when annotating targets and later correct them. However, the incorrect information may have already propagated to the Planner agent, resulting in assignments for FRs. Those assignments may have subsequently confirmed by a Silver commander before the incorrect information is discovered. In order to be in control of such dynamic situations and to effectively manage such changes, we develop a *Provenance Manager* that tracks significant information changes at ProvStore that may have impacts on decisions already made. It works as follows:

- **Listening to information invalidation**: ProvStore provides an API for external services to register to be notified when an event of interest occurs. Whenever prior information is invalidated, decisions and actions must be revised immediately. Therefore, the Provenance Agent asks ProvStore to notify it with any invalidation of existing information asserted by one of HAC-ER's components.
- **Identifying potential impacts**: Once an entity is invalidated, ProvStore sends a notification to the Provenance Manager which identifies all the dependents of the invalidated entity by querying ProvStore for the transitive closure of the wasDerivedFrom relations of the entity. If any of the dependents are confirmed decisions, the entity `confirmed_plan/178` in Fig. 8b for example, their owners will be notified next.
- **Notifying affected actors**: As each entity (e.g. a decision) is attributed to a human or agent actor, when its validity may be affected by now-invalid input(s), the actors may need to re-evaluate their decision. Hence, the Provenance Agent will send a notification to the affected actor, informing them of the fact that an entity has just become invalid, the other entities potentially affected by this, and the provenance information that has all the details for further investigation. For example, the entity `uav/target/33.1` (in Fig. 8a), may have been mistakenly identified as an "Infrastructure Damage". The error was later corrected, resulting in a new version of the target (`uav/target/33.2`) and the original version invalidated. The same chain of derivations as shown in Fig. 8b was identified by the Provenance Manager, and a notification was generated and sent to the commander who had confirmed the task allocation specified in `confirmed_plan/178`.

With the information and decision tracking provided by the Provenance Manager in tandem with ProvStore, our pilot tests have demonstrated that the system maintains situation awareness of the participants in a timely manner whenever the information used for task planning is invalidated. Silver commanders were notified of the changes in less than a minute after they occurred. More importantly, the generic provenance tracking and monitoring mechanisms implemented here are flexible and do not depend on this particular domain. Future changes in HAC-ER's individual components, or even the addition of new components, will not affect the existing operation of the Provenance Manager as long as the components report provenance information from their operation to ProvStore as previously described in Section 6.1.

7. CONCLUSIONS

In this paper, we presented HAC-ER, a prototype disaster management system based on Human-Agent Collectives. The individual components of HAC-ER demonstrate how humans and agents can be coalesced in flexible and social relationships to manage information and decisions, in order to address specific challenges faced in gathering a high level of situational awareness from crowds and UAVs, and in allocating tasks to first responders on the ground. Our prototype has been field tested with more than 100 users so far and validated by real-world emergency responders. Our field trials of the HAC-ER system have so far shown that the performance of the system is improved when greater control over autonomy is given to users (e.g., in the multi-UAV and the planner system in particular). Furthermore, emergency responders found that tracking the provenance of information from the Crowdscanner would significantly improve their confidence in information coming from crowds. The reader is referred to [13] and [4] for further details. Future work will look at deploying HAC-ER in disaster response training exercises and evaluating our interaction mechanisms and agents *in situ*.

Acknowledgements

This work was done as part of the EPSRC-funded ORCHID project (EP/I011587/1). We also thank RescueGlobal and BAE Systems for their feedback on initial versions of the system.

REFERENCES

[1] 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.

[2] M. Cummings, A. Brzezinski, and J. D. Lee. The impact of intelligent aiding for multiple unmanned aerial vehicle schedule management. *IEEE Intelligent Systems: Special Issue on Interacting with Autonomy*, 22(2):52–59, 2007.

[3] F. M. Delle Fave, A. Rogers, Z. Xu, S. Sukkarieh, and N. R. Jennings. Deploying the max-sum algorithm for decentralised coordination and task allocation of unmanned aerial vehicles for live aerial imagery collection. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 469–476. IEEE, 2012.

[4] J. E. Fischer, S. Reeves, T. Rodden, S. Reece, S. D. Ramchurn, and D. Jones. Building a bird's eye view: Collaborative work. In *Proceedings of SIGCHI (To appear)*, 2015.

[5] T. Grant. Unifying planning and control using an ooda-based architecture. In *Proceedings of the 2005 annual research conference of the South African institute of computer scientists and information technologists on IT research in developing countries*, pages 159–170. South African Institute for Computer Scientists and Information Technologists, 2005.

[6] T. Grant and B. Kooter. Comparing ooda & other models as operational view c2 architecture topic: C4isr/c2 architecture. *ICCRTS2005, Jun*, 2005.

[7] T. D. Huynh and L. Moreau. ProvStore: A public provenance repository. In *5th International Provenance and Annotation Workshop (IPAW'14)*, Cologne, Germany, 2014.

[8] N. R. Jennings, L. Moreau, D. Nicholson, S. D. Ramchurn, S. Roberts, T. Rodden, and A. Rogers. On human-agent collectives. *Communications of the ACM*, 57(12):33–42, 2014.

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

[10] K. S. Macarthur, R. Stranders, S. D. Ramchurn, and N. R. Jennings. A distributed anytime algorithm for dynamic task allocation in multi-agent systems. In W. Burgard and D. Roth, editors, *AAAI*. AAAI Press, 2011.

[11] L. Moreau and P. Missier. PROV-DM: The PROV data model. Technical report, World Wide Web Consortium, 2013. W3C Recommendation.

[12] N. Morrow, N. Mock, A. Papendieck, and N. Kocmich. Independent Evaluation of the Ushahidi Haiti Project. *Development Information Systems International*, 8:2011, 2011.

[13] S. D. Ramchurn, F. Wu, W. Jiang, J. E. Fischer, S. Reece, S. Roberts, C. Greenhalgh, T. Rodden, and N. R. Jennings. Human-agent collaboration for disaster response. *Autonomous Agents and Multi-Agent Systems: Special Issue on Human-Agent Interaction (to appear)*, 2015.

[14] C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, 2006.

[15] S. Reece, S. Roberts, D. Nicholson, and C. Lloyd. Determining intent using hard/soft data and gaussian process classifiers. In *Information Fusion (FUSION), 2011 Proceedings of the 14th International Conference on*, pages 1–8. IEEE, 2011.

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

[17] E. Simpson, S. J. Roberts, A. Smith, and C. Lintott. Bayesian combination of multiple, imperfect classifiers. In *NIPS 2011*, Oxford, December 2011.

[18] P. Smith, C. McCoy, and C. Layton. Brittleness in the design of cooperative problem-solving systems: the effects on user performance. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, 27(3):360–371, May 1997.

[19] J. Villaveces. Disaster response 2.0. *Forced Migration Review*, 38:7–9, 2011.