

# A Methodology for Deploying the Max-Sum Algorithm and a Case Study on Unmanned Aerial Vehicles

F. M. Delle Fave, A. Farinelli\*, A. Rogers and N. R. Jennings

University of Southampton, SO17 1BJ Southampton, UK

{fmdf08r, acr, nrj}@ecs.soton.ac.uk

\*Università di Verona, I-37134 Verona, Italy

alessandro.farinelli@univr.it

## Abstract

We present a methodology for the deployment of the max-sum algorithm, a well known decentralised algorithm for coordinating autonomous agents, for problems related to situational awareness. In these settings, unmanned autonomous vehicles are deployed to collect information about an unknown environment. Our methodology then helps identify the choices that need to be made to apply the algorithm to these problems. Next, we present a case study where the methodology is used to develop a system for disaster management in which a team of unmanned aerial vehicles coordinate to provide the first responders of the area of a disaster with live aerial imagery. To evaluate this system, we deploy it on two unmanned hexacopters in a variety of scenarios. Our tests show that the system performs well when confronted with the dynamism and the heterogeneity of the real world.

## 1 Introduction

Current research in artificial intelligence has dedicated considerable effort to developing technologies for disaster management. Some of these seek to help first responders to quickly assess the area of a disaster; thereby providing situational awareness—the ability to make sense of, and predict, what is happening in an environment (Endsley 2000). In these settings, the deployment of autonomous unmanned aerial or ground vehicles (UAVs and UGVs), has been advocated, since these are capable of gathering such information in an efficient and timely fashion, without relying on valuable and scarce human resources to control them (Bethke, Valenti, and How 2008).

However, to do so, it is necessary for these vehicles to coordinate their decision making to ensure that they perform effectively as a team, as opposed to acting independently. To achieve this, a variety of algorithms have been produced, among which decentralised algorithms are typically preferred since they are scalable and robust to component failure (Bethke, Valenti, and How 2008; How et al. 2009). In such cases, the key challenge is to produce techniques that allow these vehicles to make joint decisions within the computation and communication constraints of the system. Examples are many, from market-based algorithms (Dias et al. 2006), to algorithms inspired by game theory (Bourgault,

Furukawa, and Durrant-Whyte 2004) and constraint optimisation (Fitzpatrick and Meertens 2003). Among these, the max-sum algorithm, a message passing algorithm relying on the generalised distributive law (GDL) (Aji and McEliece 2000), has been shown to yield the most efficient decisions on a variety of simulated problems, whilst being scalable, robust and requiring very little computation and communication (Rogers et al. 2011).

However, despite its potential, thus far max-sum has not been deployed on a real system. It has only been tested in simulation, which lacks the dynamism and the heterogeneity of the real world. Hence, max-sum’s robustness and flexibility for real applications have not been tested in practice. Moreover, max-sum’s performance depends greatly on the way it is applied to a problem. In fact, this performance is affected by both the way in which a generic problem is encoded to form the input of the algorithm, and the way the algorithm is decentralised between the available sources of computation. However, whereas a variety of ways to apply the algorithm have been described by literature, a general framework that discusses and analyses these issues is absent.

To address these shortcomings, this paper presents a study of the deployment of the max-sum algorithm. First, we introduce a methodology that the less-experienced developer can use to deploy max-sum for problems related to situational awareness. In so doing, we describe a set of general rules that unify the different ways in which the algorithm can be applied to these problems and analyse their advantages and disadvantages. Second, we present a case study whereby we apply our methodology to deploy max-sum to coordinate a team of unmanned aerial vehicles to provide live aerial imagery to the first responders operating in the area of a disaster. In so doing, we propose a potential system that could be effectively deployed for real world operations. We then evaluate the performance of the algorithm by deploying it on two hexacopter UAVs and performing live flight tests in a number of different settings. In so doing, we make the following contributions to the state of the art:

- We develop a methodology to apply max-sum to problems related to situational awareness<sup>1</sup>. In so doing, we unify the various existing techniques and organise them into a

<sup>1</sup>The algorithms may be applicable to other domains with similar characteristics. This will be considered for future work.

sequence of steps that can guide the less experienced developer to the successful deployment of the algorithm.

- We present a potential system for disaster management, in which first responders interact in real time with unmanned aerial vehicles to request live imagery of the area of a disaster.
- We evaluate the system by deploying it on two hexacopter UAVs in three different settings. These tests demonstrate its flexibility, and therefore they show that max-sum is a strong candidate to be deployed on real unmanned vehicles for problems related to situational awareness.

The remainder of this paper proceeds as follows: Section 2 introduces max-sum; Section 3 presents our methodology; Section 4 presents the case study; Section 5 describes the flight tests and Section 6 concludes.

## 2 The Max-Sum Algorithm

Here we briefly describe the max-sum algorithm (refer to (Rogers et al. 2011) for more details). This algorithm solves problems defined by a set of variables  $\mathbf{x} = \{x_1, \dots, x_M\}$  defined over a set of discrete domains  $\mathcal{D} = \{D_1, \dots, D_M\}$  and a set of  $\mathcal{U} = \{U_1, \dots, U_N\}$ , where each  $U_j \in \mathcal{U}$  is defined over a subset  $\mathbf{x}_i \in \mathbf{x}$  of the set of variables. The output is a joint variable assignment  $\mathbf{x}^*$  such that the sum of all the functions is maximised:

$$\mathbf{x}^* = \arg \max_{\mathbf{x}} \sum_{i=1}^N U_i(\mathbf{x}_i) \quad (1)$$

To apply the algorithm to solve Equation 1, the problem is encoded into a factor graph; an undirected bipartite graph whose vertices represent variables  $x_j$  and functions  $U_i$  and the edges the dependencies between them. The algorithm then works by propagating messages between the functions and the variables of the factor graph. These messages take two forms, messages sent from a variable  $x_j$  to a function  $U_i$ , denoted by  $q_{j \rightarrow i}(x_j)$  and messages from function  $U_i$  to variable  $x_j$ , denoted by  $r_{i \rightarrow j}(x_j)$ . The former are calculated by Procedure 1, while the latter are calculated by Procedure 2. Note that each message is a function of the corresponding variable. In problems where the structure does not change (i.e. the functions and variables of a problem remain the same), such as the graph colouring problem or the scheduling problem, these procedures are typically run for a pre-defined number of iterations (proportional to the number of nodes of the factor graph). However, in most real world settings the structure of the problem varies continuously to incorporate the changes of the environment. Indeed, in these settings, the number of the variables  $x_j$  and of the utilities  $U_i$  can change as can the variables' domains and the utilities' values. To deal with this, various approaches have been defined, which will be discussed in the next section.

The propagation of these messages allow the algorithm to compute for each variable  $x_j$  the marginal function  $z_j(x_j) = \max_{\mathbf{x} \setminus \{x_j\}} U(\mathbf{x})$ . This function calculates the dependency between  $x_j$  and the global function  $U(\mathbf{x})$ . Max-sum then calculates the best assignment of each variable

$x_j^* = \arg \max_{x_j} z_j(x_j)$  as the sum of the messages flowing into  $x_j$ :

$$x_j^* = \arg \max_{x_j} \sum_{i \in M(j)} r_{i \rightarrow j}(x_j) \quad (2)$$

where  $M(j)$  is the set of all the function neighbours of  $x_j$ .

---

**Procedure 1** *messageToFunction*( $x_j, U_i, \mathbf{R}$ ): The procedure to compute a message from variable  $x_j$  to function  $U_i$ .

**Input:**  $x_j$ : the sending variable;  $\mathbf{R}$ : the set of all the messages received by  $x_j$  since the last time a message was computed;  $U_i$ : the destination function.

**Output:**  $q_{j \rightarrow i}(x_j)$   
1:  $q_{j \rightarrow i}(x_j) = \mathbf{0}$   
2: **for all**  $r_{k \rightarrow j} \in \mathbf{R}; k \neq i$  **do**  
3:    $q_{j \rightarrow i}(x_j) = q_{j \rightarrow i}(x_j) + r_{k \rightarrow j}(x_j)$   
4: **end for**  
5: **return**  $q_{j \rightarrow i}(x_j)$

---



---

**Procedure 2** *messageToVariable*( $x_j, U_i, \mathbf{Q}$ ): The procedure to compute a message from function  $U_i$  to variable  $x_j$ .

**Input:**  $x_j$ : the destination variable;  $\mathbf{Q}$ : the set of all the messages received by  $U_i$  since the last time a message was computed;  $U_i$ : the sending function;

**Output:**  $r_{i \rightarrow j}(x_j)$   
1:  $r_{i \rightarrow j}(x_j) = -\infty$   
2: **for all**  $\mathbf{d}_i \in \mathbf{D}_i$  **do** {all the joint assignments of  $\mathbf{x}_i$ }  
3:    $\sigma = U_i(\mathbf{d}_i)$   
4:   **for all**  $d_k \in \mathbf{d}_i, (k \neq j)$  **do**  
5:      $\sigma = \sigma + q_{k \rightarrow i}(d_k)$  { $q_{k \rightarrow i} \in \mathbf{Q}$ }  
6:   **end for**  
7:    $r_{i \rightarrow j}(d_j) = \max(r_{i \rightarrow j}(d_j), \sigma)$  { $d_j \in \mathbf{d}_i$ }  
8: **end for**  
9: **return**  $r_{i \rightarrow j}(x_j) + \alpha_{ij}^2$

---

## 3 The Methodology

Our methodology is composed of five steps. Each describes one aspect to take into account when deploying max-sum.

**Step 1 – Defining Variables:** Each variable of the factor graph represents a vehicle's decisions. These decisions take one of two forms in situational awareness domains:

- **Decisions as actions:** Each action is specific to the type of vehicle that is deployed. Each action is related to the manoeuvres that the vehicle can make and is defined by discretising its motion space. For instance, the actions of a fixed wing UAV are defined as the set of bank angles that it can follow, whereas the actions of a UGV are defined as its steering inputs.
- **Decisions as tasks:** Each task is a unit of work to be attended. Examples related to disaster management include imagery requests or tracking targets such as drifting life-rafts and ground vehicles.

---

<sup>2</sup>Here  $\alpha_{ij}$  is a normalising constant that prevents the messages from becoming arbitrarily large in cyclic factor graphs (Rogers et al. 2011).

The developer needs to be extremely careful when defining variables, since their number and the size of their domain influence the performance of max-sum in two ways. First, in terms of communication overhead, the length of a max-sum message to or from a variable  $x_j$  is linear in the size of its domain ( $O(|D_j|)$ ). Second, the complexity of computing a message from a function to a variable message is  $O(\prod_{x_k \in \mathbf{x}_j} |D_k|)$ . Thus, it is polynomial in the size of the individual domains, and exponential in the number of connected variables. The latter can be problematic when decisions represent tasks since their number is likely to be large. To address this shortcoming, there exists a variant of max-sum tailored to task assignment problems that reduces the communication and computation costs required (see (Macarthur et al. 2011))<sup>3</sup>.

**Step 2 – Defining Functions:** Each function of the factor graph quantifies the impact of a joint set of decisions on the value of the objective function (Equation 1). These functions take one of two forms for situational awareness domains:

- **Utility of a vehicle:** This utility identifies the contribution of each vehicle to the set of measurements that the team is making. This representation is used when vehicles are coordinating to make collective measurements of some specific feature such as the temperature of a building, or the position of a drifting life-raft (Bourgault, Furukawa, and Durrant-Whyte 2004). In these settings, actions represent a vehicle’s decisions and are used to identify the local measurement corresponding to each of its possible manoeuvres. A collective measurement is then defined as the union of all the vehicles’ local measurements.
- **Utility of a task:** This utility quantifies the assignment of one or more vehicles to a task (e.g. providing imagery of an area or tracking a drifting life-raft). A decision then represents the assignment of a task to a UAV. Thus, a utility assigns value to determine which vehicle is more capable to attend a specific task, given the vehicle’s properties such as its battery life or current position, and the task’s properties such as its duration and importance.

**Step 3 – Allocating Nodes:** The computation related to the functions and the variables of the factor graph (procedures 1 and 2) needs to be allocated to one of the available sources of computation. These include vehicles, laptops, desktops and personal digital assistants (PDAs). This is extremely useful since UAVs and UGVs have heterogeneous and limited computational resources and, most importantly, they can fail. Each node is then assigned as follows:

- **Variable Allocation:** Each variable is allocated to the vehicle whose decisions it represents. Another option is to allocate the variable to an independent platform such as a laptop or a PDA, considering that a very reliable communication channel needs to be built between the variable and the actual vehicle in order to send each decision.
- **Function Allocation:** Each function is allocated depending on which of the two approaches defined in step 2 is used:

- **Utility of a vehicle:** Each function is allocated to the vehicle whose utility it represents.
- **Utility of a task:** Each function can be allocated to any vehicle that can attend the task. Any allocation mechanism can be used to select one of these vehicles. For instance, the one with the lowest id can be chosen.

Note that, similarly to variables, each function can be allocated to an independent platform provided that a reliable communication channel is built.

**Step 4 – Selecting a Message-Passing Schedule:** A schedule for computing procedures 1 and to make a decision (Equation 2) is necessary for three reasons. The first two have been discussed in Section 2. First, max-sum requires the nodes to share a specific number of messages before Equation 2 can be calculated for each variable (i.e. before the vehicles can compute a decision). Second, in most real world settings, the structure of the problem changes continuously, therefore messages need to be shared to incorporate and propagate the changes in the environment. Third, communication is lossy in many real world settings, therefore sending redundant messages (i.e. computing procedures 1 and 2 more frequently) can result in more messages being received. Three schedules can be used:

- **Synchronised schedule:** A node waits to receive all the messages from its neighbours before computing new ones or making a decision. This schedule can only be used in settings where communication is perfect (i.e. in simulation). When this is not the case, as in most real world settings, a node can wait for a message to arrive for an undefined amount of time. Thus it would generate a deadlock which would prevent all the other nodes from functioning.
- **Periodic schedule:** A node computes its messages (and makes a decision) periodically given the most recent messages. This schedule is highly recommended in scenarios where communication is lossy (i.e. in most real world settings).
- **Response schedule:** A node sends a new message (and makes a decision) in response to the arrival of a single message from another node. This drastically reduces the computation of redundant messages compared to a periodic schedule. Indeed, new messages are computed only in the presence of new information. However, since this redundancy is now lost, this schedule is suitable in domains where communication is not lossy.

**Step 5 – Updating the Neighbourhood:** The vehicles used in disaster management typically use broadcasting to communicate with each other, since it is cheaper and easier to implement than point to point or multicast communication. However, recall from Section 2 that each node of the factor graph sends and receives messages from a specific set of neighbours. Moreover, due to the dynamism, these neighbours change since the structure of the factor graph varies continuously. Hence, each node needs to keep track of, and constantly update, its neighbours to identify its own messages among those that it receives. Depending then on the type of node, this update happens as follows:

<sup>3</sup>In this work, the algorithm is used on a task assignment problem. Hence, this version is the one adopted here.

- **Variable nodes:**

- **Decisions as actions:** For each vehicle, the neighbouring function nodes of its decision variable are the utilities of all the other vehicles that can make measurements that overlap with its own. In order to calculate these neighbours, all the measurements collected thus far are, initially, fused into a global estimate of a feature of interest, such as the temperature of a building, or the radiations emanating from a power plant (Bourgault, Furukawa, and Durrant-Whyte 2004). Then, this estimate is used to calculate each vehicle’s contribution to the next collective measurement (Stranders et al. 2010).
- **Decisions as tasks:** For each vehicle, the neighbouring function nodes of its decision variable are the utilities of all the tasks that it can attend. These are continuously updated by the communication between the different platforms, so that completed tasks are deleted while new ones are added.

- **Function nodes:**

- **Utility of a vehicle:** For each vehicle, the neighbouring variable nodes of its utility are the decision variables of all the other vehicles that can make measurements that overlap with its own. These are calculated when updating the utility with new overlapping measurements.
- **Utility of a task:** For each task, the neighbouring variable nodes of its utility are the decision variables of all the vehicles that can attend it. These are continuously updated by the communication between the different platforms so that new vehicles that can attend the task are added, while those that cannot attend it anymore are removed.

In order to understand the way in which this methodology can be used to deploy max-sum in real world settings, we will now outline a case study in which the algorithm is used to coordinate teams of UAVs for disaster management.

## 4 The Disaster Response Case Study

In this section, we describe our disaster response system that allows first responders to request imagery collection tasks to a team of UAVs flying above the area of a disaster.

### 4.1 Problem Description

The UAVs involved in our problem are rotary wing UAVs. These are chosen because they have a wide range of motion capacities (i.e. being able to take off and land vertically, hover, fly forward, backwards and laterally) that make them very suitable for collecting aerial imagery.

The first responders request imagery collection tasks from the UAVs using a touch screen personal digital assistant (PDA). Each task  $T_i$  represents a location (in geographic coordinates) for which imagery is required. To submit a task, each first responder sets three properties (Figure 1(a)): (i) priority  $p_i = \{normal, high, very\ high\}$ , representing the importance of the task (i.e. collecting imagery of an occupied building is more important than doing so for an empty one); (ii) urgency  $u_i = \{normal, high, very\ high\}$



Figure 1: The PDA’s interface.

used to prevent tasks’ starvation, as will be discussed shortly and (iii) duration  $d_i$ , which defines the interval of time for which imagery needs to be collected. Note that a first responder does not know this duration with precision since it depends on the specific reason for which imagery is required (e.g. to search for a casualty or to check access to an area). Thus, three estimates are considered ( $d_i = \{5\ min, 10\ min, 20\ min\}$ ). To complete a task, a UAV needs to fly to the specified location, station itself above it and stream live video to the PDA until the first responder indicates that the task is completed (Figure 1(b)). The information about each submitted task is then broadcasted by the corresponding PDA, so that the UAVs in the surrounding area that receive it add the task to the set of tasks that they can potentially attend. The UAVs then *jointly* decide which task each vehicle should complete and, in so doing, they maximise the number of completed high priority tasks. We achieve this coordination by applying max-sum as discussed next.

### 4.2 Application of the Methodology

The max-sum algorithm is used here to allow the UAVs to *jointly* decide which task each of them should attend. The algorithm is deployed as follows:

**Step 1 – Defining Variables:** Each variable  $x_j$  takes values in the set of tasks that UAV  $j$  can attend. This choice follows from step 1 of the methodology, when decisions represent tasks. As a consequence, the domain of  $x_j$  is defined as the set of tasks  $\mathcal{T}_j$  that UAV  $j$  can attend, where  $\mathcal{T}_j \subseteq \mathcal{T}$  and  $\mathcal{T} = \{T_1, T_2, \dots, T_N\}$  is the set of all the submitted tasks. Since UAVs have limited communication capacities, they will only be able to attend a subset  $\mathcal{T}_j$  of the tasks in  $\mathcal{T}$ . Again this follows from step 1 of Section 3.

**Step 2 – Defining Functions:** A utility function  $U_i$  measures the utility of one or more UAVs attempting to complete task  $T_i$ . This choice follows from step 2 of the methodology applied to the case where utilities quantify the assignment of a task. In order to derive  $U_i$ , we assume that the task comple-

tion is a Poisson process<sup>4</sup> (Chitale 2008) measured over the time interval in which one or more UAVs can station itself above task  $T_i$ . To define this interval, consider *all* the UAVs that can attend  $T_i$ . Live imagery can then be collected from  $t_1 = \min_j t_j^1$  to  $t_2 = \max_j t_j^2$ , where  $t_j^1 = \frac{d_{ji}}{v_j}$  is the time<sup>5</sup> required by a UAV  $j$  to reach task  $T_i$  and where  $t_j^2 = t + b_j$  is the remaining time that UAV  $j$  can remain on station ( $b_j$  is the UAV battery capacity, in terms of remaining flight time) above  $T_i$ . The utility  $U_i$  is then defined as follows:

$$U_i(\mathbf{x}_i) = p_i \cdot u_i^{t-t_i^0} \cdot \left[1 - e^{-\lambda_i \cdot (t_2-t_1)}\right] \quad (3)$$

where  $t$  is the current time, and  $p_i$ ,  $u_i$  and  $t_i^0$  are, respectively,  $T_i$ 's priority, urgency and activation time.

Intuitively, the utility defined by Equation 3 measures the impact of each possible assignment  $\mathbf{x}'_i \in \mathbf{x}_i$  of the UAVs aware of  $T_i$  on its completion (i.e.  $t_1$  and  $t_2$  are calculated over those UAVs where  $x_j = T_i$ ). Each possible assignment then represents a different subset of the UAVs that can attend  $T_i$  and for which  $U_i$  needs to be calculated. In so doing, this utility allows the UAVs to make a variety of sophisticated decisions based on all the possible constraints of the problem. For instance, the platforms will always choose tasks with higher priority (due to the factor  $p_i$  in Equation 3). If these have same priorities, the UAVs will always choose the one that has remained unattended for a longer interval of time (due to the factor  $u_i^{t-t_i^0}$  in Equation 3)). In addition, multiple UAVs may attend a task if this extends the time span for which at least one UAV is on station above the task (due to the factor  $1 - \exp^{-\lambda_i \cdot (t_2-t_1)}$ ).

**Step 3 – Allocating Nodes:** Each UAV is allocated the variable representing its decisions. Similarly, each PDA is allocated the utility functions of the tasks submitted by the first responder. By doing so, as discussed in step 3 of Section 3, the responsibilities delegated to the platforms are reduced and the sources of computational power are used in a more efficient way. Figure 2 shows an example of a factor graph resulting from this allocation. The figure shows two UAVs (UAV<sub>1</sub> and UAV<sub>2</sub>) controlling two variables  $x_1$  and  $x_2$  and two PDAs: PDA<sub>1</sub> controls two tasks  $T_1$  and  $T_2$  (and the corresponding utilities  $U_1$  and  $U_2$ ) while PDA<sub>2</sub> controls one task  $T_3$  (and its utility  $U_3$ ).

**Step 4 – Selecting a Message-Passing Schedule:** We use a periodic schedule to compute the max-sum messages. This choice follows from step 4 of Section 3. Each device (UAV or PDA) then periodically runs procedures 1 and 2 to compute the new messages and decisions (Equation 2), given the messages that it received.

**Step 5 – Updating the Neighbourhood:** Within our setting, variables can appear and disappear at anytime since UAVs can run out of battery and new tasks are constantly submitted or completed. Thus, as suggested by step 5 in Section 3, UAVs and PDAs continuously share information

<sup>4</sup> $\lambda_i = \frac{1}{a_i}$  is the rate parameter of the Poisson process.

<sup>5</sup> $d_{ji}$  is the Euclidean distance between a UAV  $j$  and task  $T_i$  and  $v_j$  is UAV  $j$ 's average speed

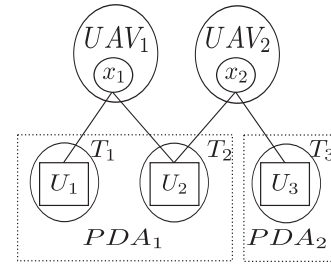


Figure 2: A factor graph showing 2 variables nodes, 3 function nodes and the platforms controlling them.

about their status (e.g. their positions, the UAVs' remaining battery and the tasks' properties). Then, the information about each variable's neighbours is stored and updated as new tasks are submitted or completed. Similarly, the information about each function's neighbours is updated each time a new UAV becomes able to complete the function's task or runs out of battery life.

## 5 Flight Tests

To ascertain the performance of max-sum when deployed on real vehicles, we deployed our system on two unmanned autonomous hexacopters over three different settings. We used two commercial off-the-shelf Mikrokopter hexacopter rotary wing UAVs<sup>6</sup> (Figure 3). Each vehicle uses a waypoint-based guidance system to control its motion—it follows a sequence of waypoints representing locations to reach. A miniature video camera provided aerial imagery. Our tests were run at a test facility outside of Sydney, Australia. A video summarising the tests can be found at <http://vimeo.com/34800379>. In the video (see Figure 4 for a snapshot), windows *A* and *B* show the hexacopters, window *C* shows the computation over the factor graph over which max-sum is running and window *D* shows the path of the UAVs. We conducted three tests:

**Flight 1 – Homogeneous Tasks:** Two identical tasks (normal priority and urgency, 5 min duration) are simultaneously submitted to the UAVs. The aim of this is to test the behaviour of the system in response to a canonical coordination scenario. In this setting, the maximum of each task's utility is obtained when the task is assigned to the closest UAV (this is due to the exponential factor in Equation 3). Thus, the coordinated decision that maximises Equation 1 is the one in which each UAV is assigned a single task. Indeed, this is what we observed during our test, confirming the correctness of our system.

**Flight 2 – Sequential arrival of Tasks:** Two different tasks (one with a normal and one with a high priority, both have normal urgency and 5 min duration) are submitted to the UAVs. One task is submitted 40s after the other. The aim of this is to test the behaviour of the system in the presence of heterogeneous properties and dynamism. Initially, one single task is present and the maximum of its utility is

<sup>6</sup>To avoid collisions, the UAVs were flown at separate altitudes (20m and 40m).



Figure 3: The “Hexacopter” UAVs used in the flight tests presented in Section 5.

obtained when it is assigned to both the UAVs (due to the exponential factor in Equation 3). As soon as the new task appears, the setting becomes the same as per flight 1. Thus, the maximum of each task’s utility is obtained when the task is assigned to the closest UAV. Two coordinated decisions then maximise Equation 1. Initially, the best decision is the one in which both the UAVs are assigned to the only available task. Then, the best decision becomes the one in which each UAV is assigned a single task. Again, this is what we observed during our test.

**Flight 3 – Heterogeneous Tasks:** Two identical tasks (normal priority and urgency, 5 min duration) are submitted to the UAVs. However, here, one UAV receives the information about one single task, while one receives the information about both. After 60s a new task (same properties as the previous ones) is submitted to both the UAVs. The aim of this is to test the behaviour of the system when the capabilities of the UAVs are heterogeneous. Initially, only one assignment is possible since one UAV can only attend one task. Thus, the maximum of this task’s utility is obtained when the former UAV is assigned to it (Equation 3). The same applies for the remaining vehicle and task. As soon as the new task appears, as per flight 1, the maximum of its utility is obtained when it is assigned to the closest UAV. Thus, two coordinated decisions maximise Equation 1. Initially, the best decision is the one that assigns each UAV to a single task. However, as soon as one UAV completes its task, the best decision becomes the one in which this UAV is assigned to the new task.

## 6 Conclusions and Future Work

We have presented a methodology that provides the first systematic framework to identify the choices that need to be made to apply the max-sum algorithm to problems related to situational awareness. We then presented a case study in which the methodology is applied to deploy max-sum on a system that allows first responders to interact with a team of UAVs to collect live aerial imagery of the scene of a disaster. Next, we deployed our system on two unmanned autonomous hexacopters over a variety of different settings, in order to evaluate max-sum’s performance when deployed on real vehicles. These tests indicated that the system performs well when confronted with the dynamism and the heterogeneity of the real world. Thus, they helped validate our methodology and confirmed that max-sum is a powerful technique to coordinate teams of unmanned vehicles for disaster management.

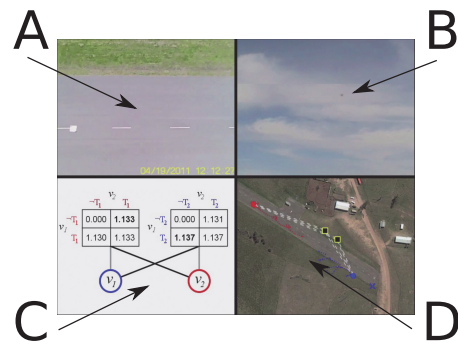


Figure 4: A snapshot of the video summarising the three flight tests.

Our future work will be focused on demonstrating that the algorithm scales beyond two UAVs. Thus far this was shown in simulation but still needs to be verified on real vehicles. Obviously, the complexity of operating large numbers of UAVs simultaneously greatly increases the complexity of the flight tests. Furthermore, we intend to study the applicability of our methodology to domains different than situational awareness, such as the energy domain or grid computing, where a large number of agents needs to be considered.

## References

- Aji, S. M., and McEliece, R. J. 2000. The Generalized Distributive Law. *IEEE Trans. on Inf. Theory* 46:325–343.
- Bethke, B.; Valenti, M.; and How, J. 2008. Uav Task Assignment. *IEEE Robotics & Automation Magazine* 15(1):39–44.
- Bourgault, F.; Furukawa, T.; and Durrant-Whyte, H. F. 2004. Decentralized Bayesian Negotiation for Cooperative search. In *Proceedings of the IEEE Int. Conf. on Intelligent Robots and Systems*, 2681–2686. Sendai, Japan.
- Chitale, R. H. 2008. *Probability and Queueing Theory*. Technical Publications.
- Dias, M.; Zlot, R.; Kalra, N.; and Stentz, A. 2006. Market-based Multirobot Coordination: A Survey and Analysis. *Proceedings of the IEEE* 94:1257 – 1270.
- Endsley, M. R. 2000. *Situation Awareness, Analysis and Measurement*. Lawrence Erlbaum Associates.
- Fitzpatrick, S., and Meertens, L. 2003. Distributed Coordination through Anarchic Optimization. In *Distributed Sensor Networks*, 257–295.
- How, J.; Fraser, C.; Kulling, K.; Bertuccelli, L.; Toupet, O.; Brunet, L.; Bachrach, A.; and Roy, N. 2009. Increasing Autonomy of UAVs. *IEEE Robotics & Automation Magazine* 16(2):43–51.
- Macarthur, K.; Stranders, R.; Ramchurn, S. D.; and Jennings, N. R. 2011. A Distributed Anytime Algorithm for Dynamic Task Allocation in Multi-agent Systems. In *Proc. of AAI*, 356–362.
- Rogers, A.; Farinelli, A.; Stranders, R.; and Jennings, N. R. 2011. Bounded approximate decentralised coordination via the max-sum algorithm. *Artificial Intelligence* 175(2):730–759.
- Stranders, R.; Delle Fave, F. M.; Rogers, A.; and Jennings, N. 2010. A decentralised coordination algorithm for mobile sensors. In *Proc. of AAI*, 874–880.