Agent-based mission management for a UAV

Samin Karim [1], Clint Heinze [2], Shane Dunn [2]

[1] *Information Systems, University of Melbourne*
Melbourne, Australia, mkarim@pgrad.unimelb.edu.au

[2] *Defence Science and Technology Organisation*
Melbourne, Australia, {clinton.heinze,shane.dunn}@dsto.defence.gov.au

Abstract

This article reports on the design, implementation and testing of a mission management system for a small, electric powered UAV - the Codarra Avatar. The system was designed and implemented using the agent-programming paradigm, which is a powerful, scalable and flexible framework for building autonomous systems. Specifically, we have used the JACK Intelligent Agents programming language. The system was integrated with an existing autopilot and auto-stabilisation system that performs basic flight control on the UAV. The mission management system performs without any human intervention, and represents a step forward in UAV autonomy. The use of agents as the design and development platform is particularly useful in view of future work involving teams, or perhaps swarms, of UAVs.

Two agent designs inspired by different design principles will be described and compared. The simulation and actual flight tests will also be discussed. In addition to the experimental report, we provide a short conceptual discourse on autonomous agent and multi-agent systems, and how the proposed frameworks and concepts can apply to multi-UAV control.

1. INTRODUCTION

UAV technology has progressed steadily over recent years to include highly sophisticated control and artificially intelligent capabilities [11]. The Predator UCAV [13], [14], Global Hawk [14] and Pegasus [6] UAVs/UCAVs all represent major advances in this field. Up to now, most UAV operations consist of large teams of human operators that oversee and control the UAV's actions. One of the aims of this research is to explore methods of reducing UAV reliance on human operators, thus making UAVs a more economical and scalable option. This paper reports on design and conceptual developments for a simple UAV path planning mission.

The UAV was given the task of independently deciding between two alternate paths to take depending on data that it has collected via its sensors. The mission management system (MMS) we have designed, developed and flight tested is different from most other similar systems in that there is no human intervention in the decision making processes. The system processes and reasons over sensory data and issues high level commands that are executed by the Flight Control System (FCS). The system was developed in the agent-oriented paradigm. *Agents* are the central building block of agent-based systems. Akin to objects in the object-oriented software engineering paradigm, they are generally described as computer systems with two important and distinguishing capabilities [18], [17]. Firstly, they are capable of *fully autonomous behaviour*, which entails independent reasoning, decision making and action in order to satisfy the agents' assigned goals. Secondly, they are *situated in an environment* which contains other agents with which they can interact by way of social protocols such as coordination, cooperation, negotiation, etc.

The agent programming language that the system was implemented in was *JACK*[1]. JACK is a powerful language that is used to develop open systems with complex and adaptive behaviour. *JACK Teams* is a programming extension of JACK that is used for developing agents with requirements for coordination and other types of team interactions. Although the system presented here is a single agent, the concepts and designs are readily extendible to a teams context for future projects.

2. MULTI-AGENT SYSTEMS

The idea of using agents for mission management and planning is particularly interesting as a basis for distributed architectures and multiagent systems (MAS). In cooperative MAS, multiple agents work together to achieve one or more desired common goals. The overall system goal is achieved through interactions and coordination of the individual agents [17]. A distributed mobile robotic team (in mobile robotic parlance referred to as a *robot collective*) has advantages over a single, complex robot in many applications [7]. For example, for search and rescue operations, multiple robots can forage far more effectively than a single, complex robot [2].

Swarm Intelligence is the study of collective intelligence, originally exhibited by large groups (or *swarms*) of animals, usually insects such as ants and bees. It is an example of team coordination in MAS. In 'robotic swarms', the agents are designed to be simplistic (in terms of cognitive decision making and sensory processing abilities), and are *unaware*

[1] http://www.agent-software.com

of their organisational position within a collaborating team. However, the system intelligence *emerges* from the interactions of these simple agents [3]. Vincent and Rubin [16] have recently presented an efficient framework and algorithm for cooperative search using UAVs based on swarm topologies and theories.

However, despite the appeal of relatively simple agents in swarm intelligence, we believe team coordination and control in multi-UAV applications cannot exist without some sort of supervisory presence or *team awareness*. Essentially, there needs to be some sort of supervision at the cognitive level, filtering the reflexive, non-cognitive swarm 'urges'. When there is a reflexive urge to fly somewhere, it needs to be tempered by many considerations before the actual decision is made to carry out the particular manoeuvre (eg. Do I need ATC (Air Traffic Control) permission?, What is my energy state?, What are the weather conditions?, etc.) An interesting research question then arises: How does such high-level supervision affect the outcomes of swarm type models in UAV, and other, team applications?

The taxonomy of a robotic collective can usually determine the best organisational and coordination framework to use for a specific domain [7]. Many interaction and coordination protocols have been developed, and a few will be mentioned here [17] as examples of effective MAS coordination frameworks:

Contract Net [15] In this protocol, a contract mechanism is used to effectively delegate tasks to the most appropriate agent. A *manager* agent broadcasts a set of tasks to other *contractor* agents within the team. The contractor agents then deliberate over the tasks outlined in the broadcast message, and 'interested' contractor agents submit a *bid* to the manager agent proposing plans or solutions to achieve the task goals. The manager then selects the bid that it perceives to yield the highest expected utility, resulting in a contractual obligation for the task to be completed by the contractor agent. This process can recurse indefinitely until an agent actually carrys out the task.

Blackboard Systems. [12], [17] In this protocol, communication is achieved by a centrally located 'database' which all agents within the system have access to. Messages to one or more agents can be delivered by flexibly entering them on the *blackboard*. Also, the state of the world can be stored by individual agents in the blackboard, resulting in an incrementally updated global view of sensory and reasoning data of the entire team that can be accessed by all team members. Another benefit of blackboard systems is that knowledge sources (KS), or agents, that contribute to the blackboard contents do not necessarily need to know the entire problem space, and hence can make a contribution based on their knowledge and/or expertise on a particular aspect of the problem. Of course, the only limitation with blackboard systems with respect to UAV teams, is that all agents must have an adequately reliable communication link to the blackboard database. This may not be possible for certain applications. Hence, with respect to this limitation, protocols that require a centralised database may not be ideal. Rather, protocols that utilise localised databases that are only accessed by relevant agents, are more desirable.

3. The Avatar UAV

The Codarra Avatar [5] is a lightweight UAV, purpose-built for small-scale, autonomous reconnaissance and surveillance missions. Codarra[2] initially perceived a need for tactical surveillance for military platoon-sized patrols. A small, programmable UAV was developed to address this need. The Avatar can be disassembled and reassembled in a short time, launched by hand (ie. thrown in the air for take-off), and transportable in a back-pack. Propulsion is via an electric motor through a fully-folding propeller that endows the UAV with gliding and powered flight capabilities. Flight endurance is approximately sixty minutes. Recovery (landing) is by parachute that deploys from a bulge above the wing centre-section. The payload-bay can accommodate anything from a video camera that is capable of transmitting video back to a ground station, to other sensors up to 1.5 kg in weight and about the size of a standard house brick. Standard sensors include: GPS receiver (with a precision of 100 metres), barometric altimeter, and a pitot-static air speed indicator. Communication range to the Ground Control Station (GCS) is approximately 10 km. Arbitrary data, as well as command signals, can be communicated to the UAV via this radio link.

A. Issues in providing the Avatar with autonomous behaviour

The issues in providing the Avatar with autonomy are:

Flight time/range As the UAV is battery powered, the relative flight time/range is small compared to, for example, a standard petrol powered radio-controlled (RC) miniature aircraft. For missions requiring longer flights, this can be achieved by 'refuelling' stops (ie. the UAV returns to base for a battery change, or another UAV 'baton changing' for re-launch).

Durability The Avatar is a lightweight, small aircraft, and therefore cannot withstand turbulent, high speed wind conditions. In certain conditions, the Avatar can overcome these situations by carefully planning flight paths. Such planning can be executed by the JACK agent mission manager onboard the UAV. However, there are situations where no amount of careful mission planning, by either the JACK agent or a human operator, can overcome the adverse conditions.

Flight regulations and restrictions Federal law requires that all aircraft pass through stringent safety checks. In particular, autonomous aircraft must exhibit *deterministic behaviour*, as certification for autonomous aircraft that can be demonstrated to exhibit deterministic behaviour is expected to be a much more straight-forward process than for non-deterministic systems. Furthermore, for mission-critical or high risk/hazard missions, deterministic behaviour is desirable. Deterministic behaviour means that the aircraft cannot be overly 'creative' and behave in a potentially random manner in known, and indeed, unknown situations. This restriction is

[2] http://www.codarra.com.au

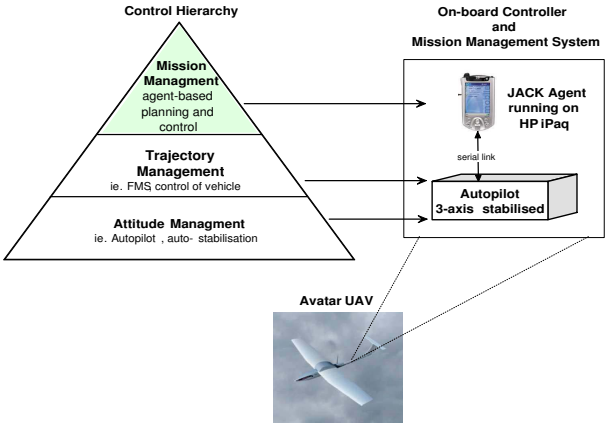

Fig. 1: Agent - FCS Architectural design [10]

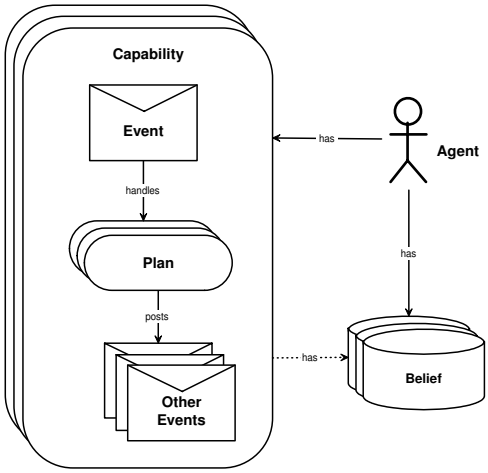

Fig. 2: An illustrated diagram of JACK agent-programming language constructs

part of the reason why we do not believe swarm topologies (refer to Section 2) are an adequate team model. The JACK programming environment that is being used with the Avatar is ideal in this regard, as the plans that it executes are known at compile time, and run deterministically in known/unknown situations.

Limited computational power As the payload of the Avatar is relatively small, the size of the agent-based Mission Management (MM) processing computer must be small and therefore may be limited in computational power. In our initial experiment, we used the *HP iPAQ* - a standard Pocket PC-based PDA (Personal Digital Assistant), which was limited both in function and in computational power. The iPAQ will most certainly be replaced by the more powerful *PC104* or similar cut-down PC platform in future test flights. A more thorough discussion follows in Section 4.

Limited Sensory Data As described at the start of this section, the Avatar is limited to GPS, altitude and speed data, with the exception of other data that it may receive from the GCS. However, for longer range missions where conditions within the vicinity of the GCS are different from that of the Avatar, this is not a practical arrangement. For such complex, long-range missions, a more sophisticated suite of sensors must be installed in the payload, such as radar, sonar or video imaging.

4. The Agent - Flight Control System (FCS) Architecture

The agent sits at the top of the control tree, issuing high level waypoint commands. The FCS then simply navigates to the specified waypoint. Data is sent to the agent at a regular, albeit slow, frequency (nominally 1 Hz). As a result, the agent must be robust and able to behave proactively in order to compensate for the slow data update rate. Refer to Figure 1 for a diagram of this architecture.

Commands are sent and an acknowledgement is anticipated by the agent. Environmental data (ie. wind speed, direction) as well as internal FCS and Ground Control Station (GCS) data are available for use by the agent.

A. Plans for the future

As mentioned above, for this inaugural test flight a simple *HP iPaq* was used as the platform for the agent. This limited the agent's functionally as well as the available computational power. However, the more powerful *PC104* platform is scheduled to be used for upcoming test flights.

5. The Agent Design

A. The JACK Agent Programming Language

JACK is an *agent-oriented programming language*. Agent-oriented languages are based around the central concept of *agents*, as described in the introduction. Architecturally, JACK consists of several constructs: Agents, Capabilities, Events, Plans and Beliefs. Agents are at the highest level of abstraction, and represent entities with the autonomous behaviour within the system. Capabilities are abstract entities that encapsulate groups of related events and plans (described below).

In JACK, multiple threads of execution are driven by separate *events*. The events are *handled* by specific *plans*, which in turn can *post* one or more other events. Autonomy is achieved by the dynamic selection of plans during run-time, depending on the specific situation or problem faced by the agent. For example, suppose an `Obstacle Detected` event was received by the agent, it could handle it with one of several plans, such as `Turn Left`, `Turn Right`, `Generate New Path` or `Emergency Stop` (if no other plan is applicable or yields the desired expected utility).

The combination of these constructs results in behaviours structured around the BDI, (**B**elief **D**esire **I**ntention) theory of agency[1], [18]. The BDI framework is based on the human rationality and goal-directed thinking processes. Each agent pursues its given goals (**desires**), adopting the appropriate plans (**intentions**) according to its current set of data (**beliefs**) about the state of the world. In this way, agents behave like humans with all the mental attributes encompassed by the BDI

framework. Refer to Figure 2 for an illustration and [1] for more information.

B. The Final Design

There were several design approaches and implementations of the MMS. We settled on an approach adopted in many simulations of military situations, called *OODA* - **O**bserve, **O**rient, **D**ecide, **A**ct. Colonel John Boyd developed the OODA model of decision making on the premise that decision making in such situations is a perpetually looping sequence of: *observing* the environment the agent is situated in, *orientating* based on these (and past) observations, *deciding* what to do, and finally executing the chosen/formulated *action*. Refer to Figures 3, 4 and 5 for JACK design diagrams of this implementation.

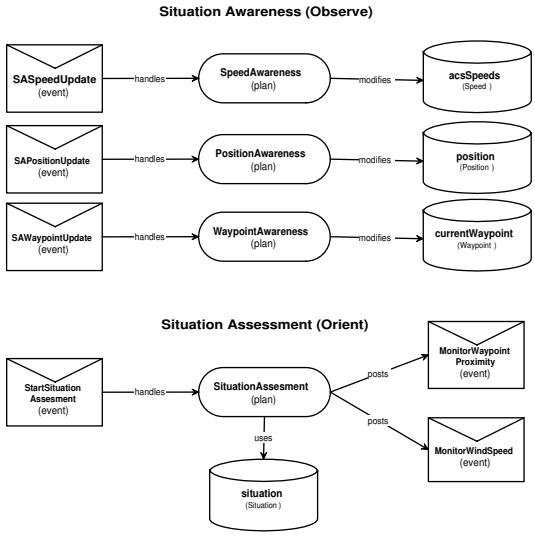

Fig. 3: OODA design approach for Mission Management System: **Observe** and **Orient**

C. An Alternate Design

An alternate design was also developed concurrently with the previously described design. It was not flight tested, despite it being functionally equivalent to the OODA approach design. However, it underwent thorough software testing and proved to be an efficient alternative to the final design.

The rationale for this particular design was a simplistic, feed-back control system. Every time a data packet arrives, the MMS either stores the relevant data, or if it is time, makes a decision based on the stored data. Refer to Figure 6 for the JACK design diagram of this implementation

6. FLIGHT TEST

DSTO's Air Vehicle Division (AVD) in collaboration with the Air Operations Division (AOD) and the Department of Information Systems, The University of Melbourne, successfully conducted a flight test of the Codarra Avatar UAV between 5 - 7 July, 2004.

Before the actual flight tests, a standard radio-controlled aircraft, similar in size and flight dynamics to the Avatar, was flown to test wind conditions and aircraft visibility. The

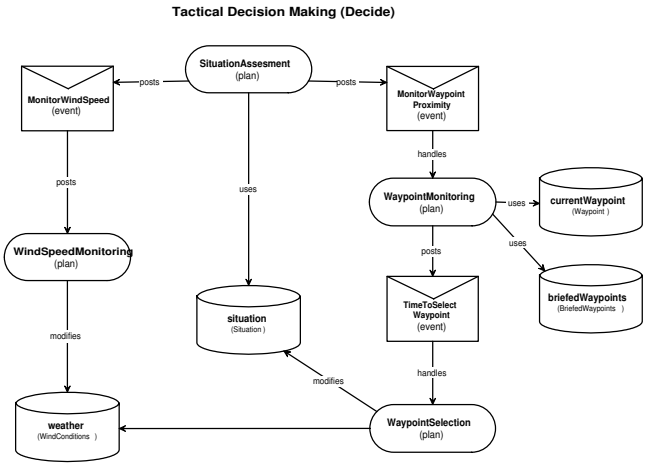

Fig. 4: OODA design approach for Mission Management System: **Decide**

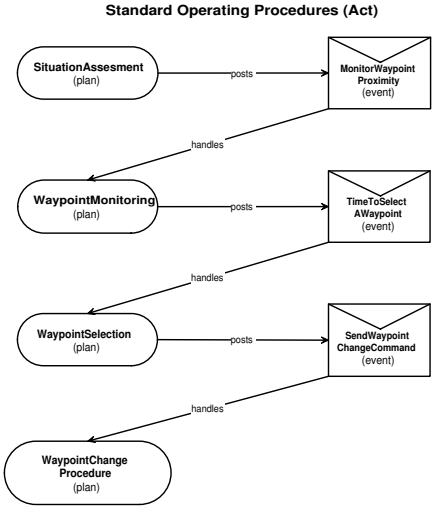

Fig. 5: OODA design approach for Mission Management System: **Act**

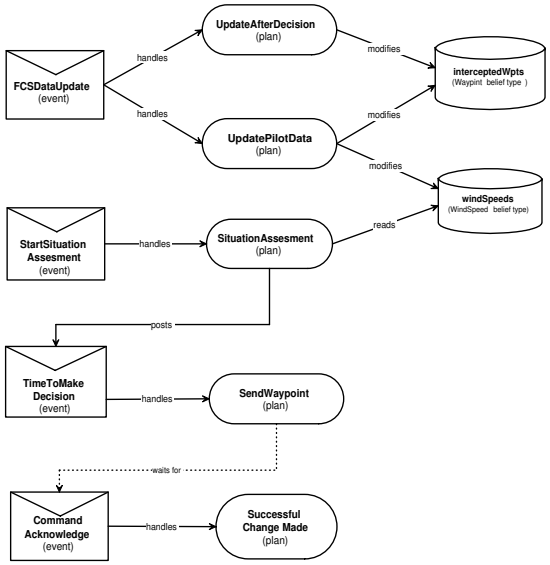

Fig. 6: Feed-back control system approach for Mission Management System

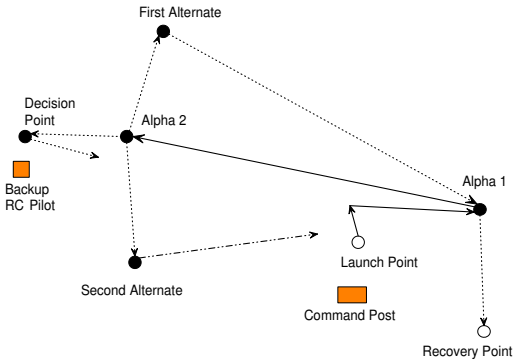

Fig. 7: Avatar UAV Flight Path: At Alpha 2 the AUC (Autonomous UAV Controller) selects one of the alternate waypoints based on the local wind conditions. Failure of the AUC will result in the aircraft flying to the decision point and then returning home

team then proceeded to configure the Avatar MMS with waypoint data and conduct a 'walk around' test. This involved, essentially, switching on the FCS and JACK agent systems, carrying the UAV around the flight course, traversing the waypoints the UAV is expected to intercept during its flight, and monitoring the FCS and JACK agent behaviour from the Ground Control Station (GCS). This testing method can be likened to hardware-in-the-loop testing.

After these tests were completed and all minor configuration issues rectified, the flight test was conducted. The UAV was hand launched and manually piloted to a stable altitude and attitude by an expert Radio-Controlled (RC) pilot. The UAV was switched over to 'UAV mode', and then proceeded to intercept the waypoints described in Figure 7. Figure 8 shows some pictures taken during the test flight.

7. Conclusions

An agent-based mission management system (MMS) for a small UAV was developed using the JACK agent-oriented programming language. The UAV has, as standard, a flight control system (FCS) that performs simple tasks such as waypoint navigation and three axis auto-stabilisation. The test mission was to make a single, fully autonomous, intelligent path planning decision mid-flight using JACK as the MMS platform, coupled to the existing FCS.

Agent-oriented languages, and particularly the JACK language, offer higher levels of autonomy, intelligence, and above all, the possibility for complex multiagent systems based teams of UAVs capable of complex interactions such as coordination and negotiation. UAV teams can achieve tasks more efficiently and flexibly than a single UAV, or can tackle problems that are not possible with a lone UAV. Contract Net and blackboard systems are examples of effective coordination mechanisms that can be used in dynamic multiagent system domains such as UAV teams.

Fig. 8: The Codarra Avatar and the test team at Greytown, Melbourne, Australia, 5 - 7 July, 2004.

8. Future Work

Future projects planned at DSTO include more complex UAV path planning tasks, and of greatest interest, an applied UAV teams project. A layered control design (eg. [8], [4]), team reasoning and planning under uncertainty (eg. [9], [19]), and team coordination and task synchronisation are among the research considerations.

Additional work involving intelligent agents and UAVs within DSTO, that extends the work presented in this paper, concerns the development of UAV systems that approach the safety and reliability of manned aircraft. In working towards this goal, agents are being developed that replicate a pilot's functionality in the management of his/her aircraft. These agents will form beliefs about the vehicle state and environment based on sensors and data from other sources. Using such data, the agents will reason appropriately in the nature of a pilot about actions in the light of the overall mission goals.

Acknowledgments

We would like to thank Agent-Oriented Software for their support, in particular David Sheperdson and Richard Jones who provided assistance during implementation. From DSTO, we would like to thank Erdal Akgol of AOD and Stephen van der Velden of AVD for their assistance in the Agent-FCS interface.

REFERENCES

[1] Agent-Oriented Software Pty. Ltd. (AOS), P.O. Box 639, Carlton South, Victoria, 3053. *JACK Intelligent Agents^{TM}: JACK Manual*, 4.1 edition, April 2003. The Main JACK Manual describing the JACK agent programming language, available at http://www.agent-software.com/shared/resources/index.html.

[2] Tucker Balch. The impact of diversity on performance in multi-robot foraging. In *Proceedings of the third annual conference on Autonomous Agents*, pages 92–99. ACM Press, 1999.

[3] Eric Bonabeau, Marco Dorigo, and Guy Theraulaz. *Swarm Intelligence: From Natural to Artificial Intelligence*. Santa Fe Institute Studies in the Sciences of Complexity. Oxford University Press, 1999.

[4] R. Peter Bonasso and David Kortenkamp. Using a layered control architecture to alleviate planning with incomplete information. In *Planning with Incomplete Information for Robot Problems: Papers from the 1996 AAAI Spring Symposium*, pages 1–4. AAAI Press, Menlo Park, California, 1996.

[5] Codarra. Codarra Avatar fact page. Web: http://www.codarra.com.au/lowlpgs/1product.html, July 2004. Includes a fact sheet: http://www.codarra.com.au/circ/cx1.pdf.

[6] John Croft. Pegasus: UCAVs look seaward. *Aerospace America*, 9:36–42, September 2003.

[7] G. Dudek, M. Jenkin, E. Milios, and D. Wilkes. A taxonomy for multi-agent robotics, 1996.

[8] E. Gat. *Artificial Intelligence and Mobile Robotics*, chapter 8: Three-Layer Architectures, pages 195–210. AAAI Press/MIT Press, 1997.

[9] Sarit Kraus, Onn Shehory, and Gilad Taase. Coalition formation with uncertain heterogeneous information. In *Proceedings of the second international joint conference on Autonomous agents and multiagent systems*, pages 1–8. ACM Press, 2003.

[10] Andrew Lucas. First Flight True UAV Autonomy At Last. *Agent-Oriented Software Press Release*, July 2004.

[11] David M. North, editor. *Aviation Week*, volume 159. McGraw-Hill, September 2003.

[12] Grantham K. H. Pang. Development of a blackboard system for robot programming. In *Proceedings of the third international conference on Industrial and engineering applications of artificial intelligence and expert systems*, pages 123–130. ACM Press, 1990.

[13] Predator. RQ-1 Predator Medium Altitude Endurance (MAE) UAV. Web: http://www.fas.org/irp/program/collect/predator.htm, November 6 2002. Includes fact sheets: http://www.af.mil/factsheets/.

[14] David L. Rockwell. Sensing the future of UAVs. *Aerospace America*, 9:26–31, September 2003.

[15] Reid G. Smith. The contract net protocol: High level communication and control in a distributed problem solver. In *IEEE Transactions on Computers*, volume C-29, pages 1104–1113, December 1980.

[16] Patrick Vincent and Izhak Rubin. A framework and analysis for cooperative search using UAV swarms. In *Proceedings of the 2004 ACM symposium on Applied computing*, pages 79–86. ACM Press, 2004.

[17] Gerhard Weiss, editor. *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*. MIT Press, 2000.

[18] Michael J. Wooldridge. *Introduction to Multiagent Systems*. John Wiley & Sons, Inc., 2001.

[19] Yang Xiang and Victor Lesser. On the Role of Multiply Sectioned Bayesian Networks for Cooperative Multiagent Systems. *IEEE Systems, Man, and Cybernetics (Part A)*, 33(4):489–501, 2003.