Wei, Yi, Gregory R. Madey, and M. Brian Blake. "Agent-based simulation for UAV swarm mission planning and execution." Proceedings of the Agent-Directed Simulation Symposium. Society for Computer Simulation International, 2013.

Agent-based Simulation for UAV Swarm Mission Planning and Execution

Yi Wei, Gregory R. Madey
Department of Computer Science and Engineering
University of Notre Dame
Notre Dame, IN 46556 USA
ywei1@nd.edu, gmadey@nd.edu

M. Brian Blake
Department of Computer Science
University of Miami
Coral Gables, FL 33124 USA
M.Brian.Blake@miami.edu

Keywords: Agent-based Simulation; UAV Swarm; Mission Planning;

Abstract

Swarms of Unmanned Aerial Vehicles (UAV) have been foreseen by multiple organizations to serve an important role in future air-based warfare and civilian operations. UAVs are less expensive than their piloted counterparts, provide greater flexibilities and remove the need for on-board pilot support. Efficient control of swarms opens a set of new challenges, such as automatic UAV coordination, efficient swarm monitoring and dynamic mission planning. In this paper, we investigate the problem of dynamic mission planning for a UAV swarm. An agent-based control framework is proposed, which employs a control agent for task assignment and multiple UAV agents for local task scheduling. A prototype simulation framework is implemented as a proof-of-concept. Experimentation with the framework suggests the effectiveness of swarm control using several mission planning mechanisms.

1. INTRODUCTION

An Unmanned Aerial Vehicle (UAV) is an aircraft which does not require an on-board pilot. The control of a UAV is usually performed either in an autonomous way by an on-board computer, or in a manual way by an operator in a remote ground station. Compared with their piloted counterparts, UAVs do not have to concern with pilot safety, hence can be made less expensive and into much smaller sizes to increase their flexibilities. Since the last decade, there has been large scale use of UAVs in various military and civilian operations, such as enemy territory reconnaissance, remote area surveillance and hazardous environment monitoring.

As UAVs' prices reduce and their capabilities improve, it is foreseen by many organizations that in the near future, swarms of UAVs will replace single ones for more complicated missions in more uncertain or even hostile environments [1]. Currently, a UAV is operated by at least one ground operator, sometimes more. This single UAV operation mode has its scalability issues as the number of UAVs in a swarm increases quickly. So, more intelligent command and control approaches and systems are required to let ground pilots "fly the swarm" instead of individual UAVs.

The autonomous nature of individual UAVs in a swarm suggests that the interaction can be modeled as a multiagent system, with communication, negotiation and coordination among different agents. In this paper, we extend our previous UAV studies that investigate the application of DDDAS (Dynamic Data Driven Application System) to UAV mission planning [3], leverage intelligent agent and workflow techniques [4], and further apply our related work for configuration and resource allocation in computing environments [5][6][7][24]. These past studies, in this paper, are applied to the swarm mission planning problem and we propose an agent-based framework for mission assignment and scheduling. The framework employs two types of agents. One is the swarm control agent and the other is the UAV agent. The swarm control agent's objective is to assign tasks in a mission to different UAV agents based on its latest knowledge about the swarm. Upon receiving the task assignment, the UAV agent tries to schedule, based on its local knowledge, the new task into its local task queue in order to minimize total task completion cost. The swarm control agent also periodically collects status information from the UAV agents to form a global view of the swarm and the environment. Such information is used to update the control agent's internal swarm model for future task assignments. To evaluate the proposed framework, a prototype simulation program is implemented.

The rest of this paper is organized as follows. In Section 2, related research work in the area of agent-based UAV swarm control and mission planning are discussed. Formal definitions of the terms used in this paper are provided in Section 3. Also in Section 3 are the descriptions of the proposed mission planning

framework. In Section 4, the architecture and implementation of the simulation program are explained. Initial experimental results are also presented. In Section 5, we summarize our work and give a brief preview of future works.

2. RELATED WORK

The new challenges imposed by UAV swarms have attracted many researchers since the last decade. New simulation models, command and control mechanisms and simulation tools have been developed to tackle issues in different aspects of the swarm. MASON [9] is a general purpose multiagent simulation library that is utilized by our previous work, along with the Matlab-based UAV simulator MultiUAV2 [10]. In [8], a swarm simulator is implemented in Java for target searching. Garcia et al. in [11] introduced a multi-UAV simulator implemented on top of a commercial flight simulator called X-Plane. And Russell et al. in [12] presented a parallel swarm simulation environment utilizing an existing parallel emulation and simulation tool called SPEEDS. Gaudiano et al. in [13] proposed an agent-based UAV model and different decentralized strategies for swarm control in search and destroy missions. A similar cooperative search problem is also discussed in [14], but in a civilian usage scenario. The problems of path planning and vehicle routing are investigated in [15] using multi-objective evolutionary algorithms. In [16], the cooperative searching problem is discussed for the purpose of detecting moving and evading targets in a hazardous environment. [17] investigates the Automatic Target Recognition (ATR) problem in UAV control and proposed a multiagent-based distributed strategy for UAV swarms. [18] proposed another agent-based model of decentralized strategy for the control of UAV swarms. The task allocation problem is discussed in [19] and [20] using different methodologies. The authors in [21] investigated the UAV communication problem using Parrot AR-Drones. In [22], the authors explored the problem of cooperating multiple UAVs for hunting a target. And in [23], different UAV command and control design strategies are discussed and compared. There are also applications of using a UAV swarm to monitor and detect wild fire hotspots [25].

The work proposed in this paper is different from previous research in that the mission planning and scheduling processes are separated to different agents, with UAV agents trying to achieve a local maximum and the control agent striving to reach a global optimization goal. This strategy can lower the burden on the controller as well as reduce potential communications between different agents.

3. AGENT BASED SWARM MISSION PLANNING

3.1. Definitions

A swarm consists of a group of UAVs. Let S = {v1, v2, ..., vn} denote a swarm, and n = |S| denote the number of UAVs in the swarm. We assume that the number of UAVs is a constant number during runtime. A UAV has the following properties:

1. Speed, currently constant speed is assumed. But different UAVs can have different speeds;
2. Maximum capacity, a UAV's capacity is some type of power source to sustain the UAV's flight, such as fuel or battery.

During a swarm's mission execution, for any given time t, each UAV has a state, defined as $v_i(t)$. A state contains the following information:

1. Current position of the UAV. We assume that all UAVs fly on the same altitude, so the current position contains x and y coordinates;
2. Current heading of the UAV;
3. Residual capacity;
4. Measured capacity consumption rate, this value along with the UAV's max capacity determine how long the UAV can stay in the operation area before having to return to base. This measurement changes overtime;
5. A list of tasks assigned to this UAV, could be empty. Here we assume that all UAVs are homogeneous, that is, a UAV can take over any task.

A swarm's state consists of two parts. One is the collection of its UAVs' states, the other is a collection of mission states for all missions scheduled on the swarm. Here we consider a long endurance swarm capable of finishing multiple missions.

A mission is a high level description of the objective that the swarm needs to accomplish. A mission consists of one or more tasks specifying the detailed steps of the mission. Let $MS = \{T, D\}$ denote the mission, where T is the set of tasks in the mission, $\{t_1, t_2, \ldots, t_k\}$, and D is set of the dependencies between different tasks. In set D, the pair $\{t_i, t_j\}$ means that task t_j has a dependency on task t_i, i.e., t_j must wait for t_i to complete in order to start. If we consider each task as a node, and the dependency relationship between two tasks as a directed edge, then a mission can be represented by a directed graph. In this paper, we consider missions that can form a Directed Acyclic Graph (DAG).

After a mission is scheduled on a swarm, there is a state associated with it for any given time t. A mission's state is a 3-tuple $MS_t = \{C, P, W\}$ where C is the set of completed tasks, P is the set of tasks that are in progress, and W is the set of tasks that are waiting to be finished. A mission is considered

completed when $P = W = \{\}$, and $C = T$. Table 1. gives a quick reference to the symbols used in this paper.

Table 1. Nomenclature

Symbol	Description
S	A UAV swarm.
v_i	The *ith* UAV in the swarm.
$v_i(t)$	The state of the *ith* UAV at time *t*.
n	Total number of UAVs in the swarm.
MS	A mission.
T	The set of tasks in the mission.
t_i	The *ith* task.
D	The set of task dependencies in the mission.
$\{t_i, t_j\}$	A task dependency pair specifying task t_j has a dependency on task t_i.
MS_t	The state of a mission at time *t*.
C	The set of completed tasks in the mission state.
P	The set of in-progress tasks in the mission state.
W	The set of not-started tasks in the mission state.
M	Scheduling mapping from the set of tasks to the swarm.

3.2. An Agent-based Framework for Swarm Mission Planning

Based on the above definitions, we consider the following problem:

Given a swarm S, either with or without existing missions, and a new mission MS, schedule the mission on the swarm so that the total mission cost is minimized.

The scheduling process produces a mapping, or an assignment, M, from the set of tasks T to the set of UAVs, S. The mapping specifies which task is assigned to which UAV. For example, $\{t_1, v_2\}$ means UAV 2 gets task 1.

The mission cost is the sum of all its tasks' costs. The cost of a task is different for different types of tasks. Generally, it consists of the travelling cost and the task completion cost. The former is the cost for a UAV to travel to the task location from its current location, while the later is the cost for the UAV to carry out the task.

To solve this problem, we propose a hybrid approach which combines global task assignment and local task scheduling. Two types of agents are used. One is the *swarm control agent* (*SCA*), and the other is the *UAV agent* (*UAV*). Usually there are one SCA and multiple UAVs in a swarm operation environment. The SCA is the interface between a swarm operator and the swarm, it usually runs in the ground station and its design objective is to decompose a newly arrived mission into multiple tasks, assign those tasks to UAVs, and monitor the status of the swarm and missions. On the other hand, A UAV agent represents an operational UAV in the swarm which is capable of completing different tasks. In the framework, agents communicate with each other through messages. These messages either require the recipient to take an action, such as the new task message, or contain the latest information about the sender, such as the status return message. Table 2. gives a complete list of all messages types used by the framework. Details of some messages are discussed below.

In order for the SCA to make task assignment decisions, it needs to first obtain the latest knowledge about the swarm and its operational environment. The SCA periodically sends status inquiries to all UAVs. Upon receiving such inquiry, a UAV generates a status message and sends it back to the SCA. The message contains the following fields:

1. Current location of the UAV;
2. Residual capacity of the UAV;
3. Measured capacity consumption rate of the UAV;
4. Estimated final location of the UAV after all of its tasks are completed;
5. Estimated final residual capacity of the UAV after completing all of its tasks.

Other than this status message, every time a UAV finishes a task, it sends a task completion message to the SCA. The SCA will use this message to update its mission execution status and assign new tasks.

Since tasks in a mission have dependencies among each other, they cannot be all assigned at the same time. So the SCA needs to decide which tasks are ready to be assigned. A *ready* task is the one that doesn't have any predecessor tasks or all of its predecessors have been finished. As a task is completed, other tasks that are dependent on it may become ready. The SCA monitors the status of each mission and assigns tasks to UAVs as they become ready. For each ready task, the SCA selects the assignment target UAV by the following steps:

1. Calculate, for each UAV, the cost of finishing the task. The SCA will use its latest knowledge about the swarm to estimate the cost of finishing it. This cost is different for different UAVs, because UAVs are on different locations and may or may not have existing tasks. There are two situations:
    a. The UAV doesn't have scheduled tasks. In this situation, the cost is the traveling cost from UAV's current location to task location plus the task's completion cost. The capacity used for comparison is the UAV's current residual capacity;
    b. There are scheduled tasks on the UAV already. In this case, the estimated final location and estimated final residual capacity values are used to calculate the task cost.
2. Sort the calculated costs and choose the UAV with the smallest cost value as the candidate for the task.

The task assignment gives a task to a UAV, but it *does not* specify how this task should be scheduled on the UAV. To be more specific, the SCA does not specify whether the new task should be executed immediately or after all existing tasks are completed. The UAV controls this decision. There are several possible scheduling policies:

1. *First Come First Serve* (FCFS): the new task will be executed after all earlier tasks are completed;
2. *Insertion based policy*: if there are no scheduled tasks on the UAV, the new task will start execution immediately. Otherwise the UAV will schedule the new task based on the following equation:

Let l_0 denote the location of UAV v_i when the new task t_m arrives. Let $T_i = \{t_1, t_2, \ldots, t_k\}$ be the set of scheduled tasks on v_i, and $L_i = \{l_1, l_2, \ldots, l_k\}$ denote the locations for scheduled tasks and l_m denote the location of task t_m. Let *os* denote the cost of the old (current) schedule, and *ns* denote the cost of the new schedule which includes t_m. Let function $D(l_i, l_j)$ denote the distance between locations l_i and l_j, and $C(t_i)$ denote the completion cost of task t_i. The cost of the new schedule, *ns*, can be calculated as:

$$ns = min(os + D(l_k, l_m) + C(t_m), min(os - D(l_i, l_{i+1}) + D(l_i, l_m) + D(l_m, l_{i+1}) + C(t_m)), 0 \le i < k \quad (1)$$

In the above equation, the first part of the outer *min* operation is the cost to place the task after all scheduled tasks. The inner *min* operation tries to find a place among the scheduled tasks to insert the new task so that the total new cost is minimized. The index *i* that produce the min total cost is the place to insert the task. If the first part of the outer min operation is smaller, then t_m will be executed after all currently scheduled tasks.

3. *Travelling Salesman Problem* (TSP): the scheduling problem can also be modelled as a TSP problem.

The nodes (cities) are the UAV's current location, locations of all scheduled tasks, and the new task's location. The cost of a link is the distance between two locations. And the starting point is the UAV's current location. Since TSP is known to be NP-hard, many heuristics have been proposed to approximately solve this problem. A simple heuristic is the nearest neighbour heuristic. At each step, the heuristic chooses the unvisited city that is nearest to the current city as the next destination.

4. *Adaptive Policy*: This policy simply uses all the other policies to calculate task costs and pick the policy that produces the minimum cost each time the UAV needs to perform a task scheduling.

A sorted task list is produced after the scheduling process, which represents the completion order of the tasks. The UAV then follows the task list to complete all tasks. In addition to the scheduling, the UAV will also update its estimated final location after all scheduled tasks (including the newly arrived task) are completed and the corresponding estimated final residual capacity. This local scheduling process is repeated every time a new task is assigned to the UAV.

Since the capacity consumption rate of a UAV varies overtime, it is possible that a UAV receives a new task then inserts the task into its current schedule, but discovers later that the schedule cannot be completed with its residual capacity. To address this situation, during the swarm operation, every UAV that has been assigned tasks periodically tests the *feasibility* of its current schedule. The feasibility of a schedule is whether the UAV has sufficient residual capacity, based on current calculated capacity consumption rate, to finish all tasks in its schedule. If the UAV decides that it is infeasible to complete the current schedule, it will remove tasks from the end of the schedule until it becomes feasible. The removed tasks are sent back in a task reassignment request to the SCA. After receiving such request, SCA will reassign those tasks to other UAVs using the same methods mentioned above.

Table 2. Communication messages used in the framework

Message Type	From	To	Description	Action on Recipient
New Mission	Users	SCA	Users send new missions to the SCA for mission planning and task assignment.	Start a mission planning process.
New Task	SCA	UAV	The SCA assigns a new task to a UAV.	Start a task scheduling process.
Status Inquiry	SCA	UAV	The SCA requests a status update from a UAV.	Generate a status return message.
Status Return	UAV	SCA	The UAV sends its latest status information to the SCA as a reply to its status inquiry.	Incorporate the latest status of the UAV into the SCA's knowledge base.
Task Completion	UAV	SCA	The UAV informs the SCA about the completion of a task.	Search for additional ready tasks and assign them to the UAVs.
Task Reassignment	UAV	SCA	The UAV requests the SCA to initialize a task reassignment for some of the tasks on the UAV.	Perform task assignment process and select new UAV for reassignment.

4. SIMULATION IMPLEMENTATION

A simulation application is built as a proof-of-concept of the proposed mission planning framework. The simulation is written as a multithread Java program, and both the SCA and all UAVs are represented by threads. Messages from Table 2. are also implemented to simulate communications between agents.

Figure 2 shows the GUI of the testbed. The left part of the GUI has the mission output area on top and the control buttons at the bottom. The testbed visualizes missions by displaying their tasks and task dependencies as DAGs. Each task node also has a color indicating the task status, shown in Table 3.

At the bottom left of the GUI are the control buttons, provided for the convenience of testbed users. The *Start* button starts a new simulation. The *Stop* button terminates a running simulation. After the simulation is stopped, the JFreeChart [26] library will be invoked to generate simulation report charts. One of these charts is shown in Figure 1. It is a pie chart showing how many tasks are completed by each UAV in the swarm. The *Pause* button in the control button area is used to pause/resume the simulation. The button text will switch from "Pause" to "Resume", and vice versa, after being clicked to indicate the intended control action.

The top area of the GUI's right part is the swarm display area. The swarm is visualized in this area, including all its UAVs, their default waypoints and all tasks scheduled on these UAVs. A UAV is represented by a small solid triangle on the GUI. The capacity of the UAV is shown by a small solid rectangle near the UAV. The color of the rectangle changes from green to yellow to red as the residual capacity diminishes. The default waypoints of a UAV are visualized by small solid squares connected by directed dashed lines. The task waypoints are also represented by small solid squares, but they are connected by undirected solid lines. The area below the swarm display area is used for textual output of simulation information, such as task assignment results, task and mission completion results, and simulation termination message.

Table 3. Task colors and their indicated task status

Task Color	Task Status
Black	Task hasn't been assigned yet.
Red	Task has been assigned to the UAV, but has not been scheduled on the UAV.
Yellow	Task has been scheduled on the UAV.
Blue	Task is currently being executed by the UAV.
Green	Task has completed.

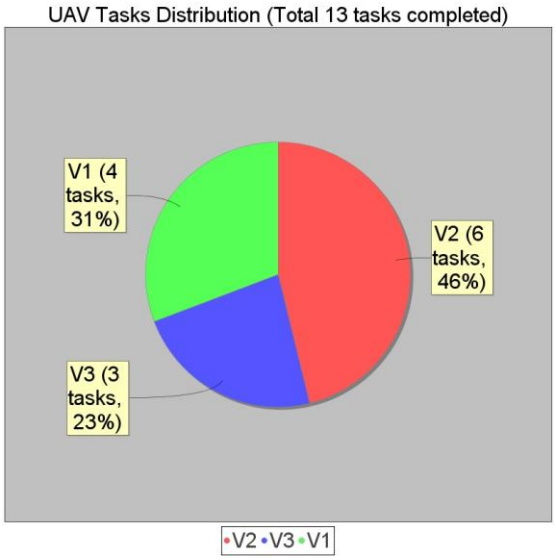

Figure 1. A pie chart generated by the testbed to report task completion status of each UAV

A set of experiments are performed to compare the effectiveness of different local task scheduling policies discussed in Section 0. Since the task scheduling process is performed by each UAV, the experiment simulated a single UAV scheduling and finishing multiple tasks on different settings. Because a scheduling policy's goal is to minimize the total costs of all scheduled tasks on the UAV, such costs are used as the evaluation metric of the scheduling policy. The completion cost of a task is static, but the travelling cost of a task differs between different policies. The total cost values of three policies (Insertion, TSP and Adaptive) are normalized against the value of the FCFS policy. Figure 3 and Figure 4 show the experimental results. The y-axis in these figures shows the relative performance of a policy compared to the FCFS policy. For example, in Figure 3, when the task count is 50, the Insertion policy produces a schedule which has a total cost equal to about 30% of the total cost of the schedule produced by the FCFS policy. In the experiments, the UAV speed is set to 5.0, and the tasks are randomly generated within a square shape area. In Figure 3, the total number of tasks increases from 50 to 500. Task arrival interval is fixed at one new task per 50 simulation steps. Unsurprisingly, all three policies (Insertion, TSP and Adaptive) performed much better than the straightforward FCFS policy. The adaptive policy performs slightly better than the insertion policy, which outperforms the TSP policy.

Figure 4 shows an experiment where the number of tasks is fixed at 100, but the arrival interval of tasks increases from 0 to 500 steps/task. We can see from

the figure that, as the arrival rate increases, relative performance of the three policies decreases, with the adaptive policy affected the least. The reason for this performance decrease is that as the task arrival interval increases, more scheduled tasks are started or even completed when new tasks arrive, making these policies more analogous to the FCFS policy.

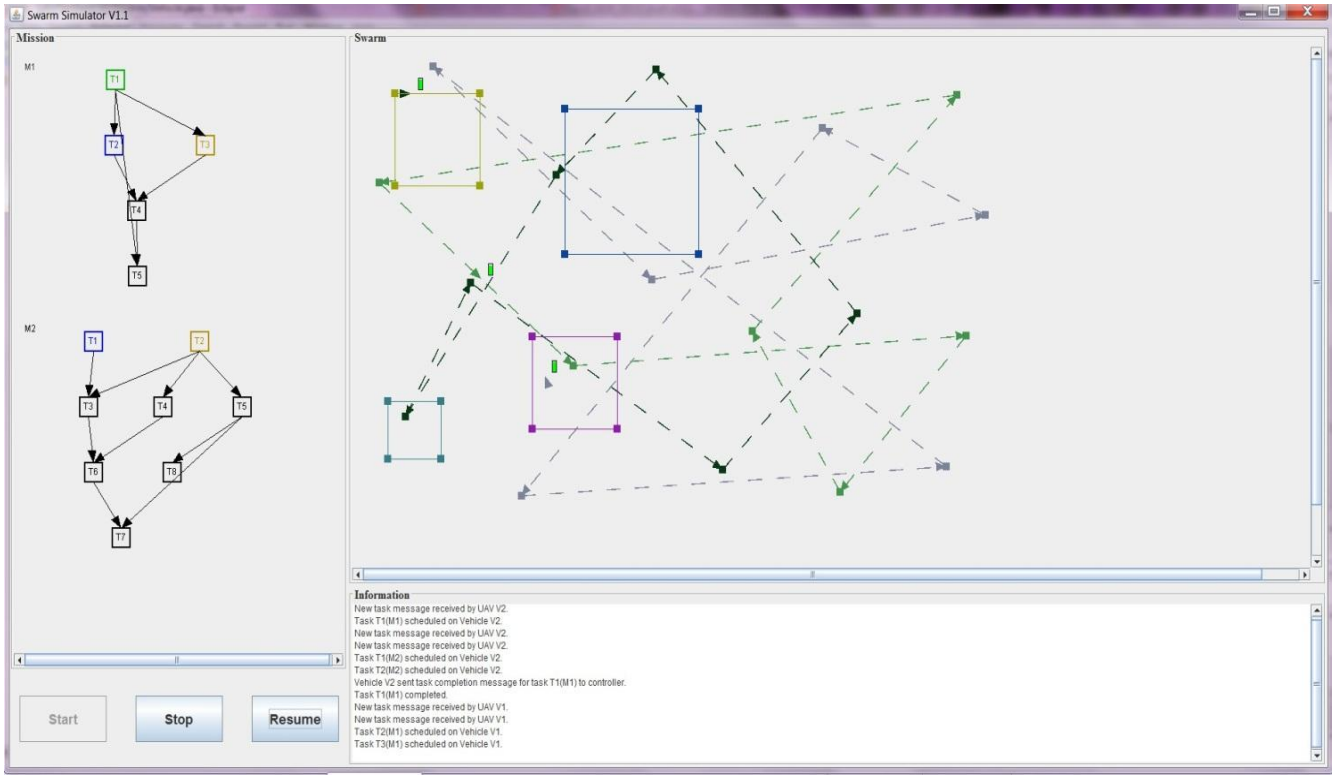

Figure 2. GUI of the simulation application showing a swarm of 3 UAVs executing 2 missions

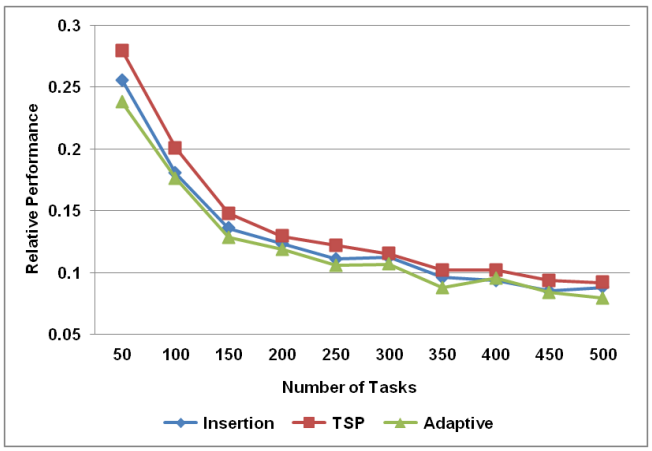

Figure 3. Relative performance of different scheduling policies compared to FCFS, with fixed task arrival interval and changing task count.

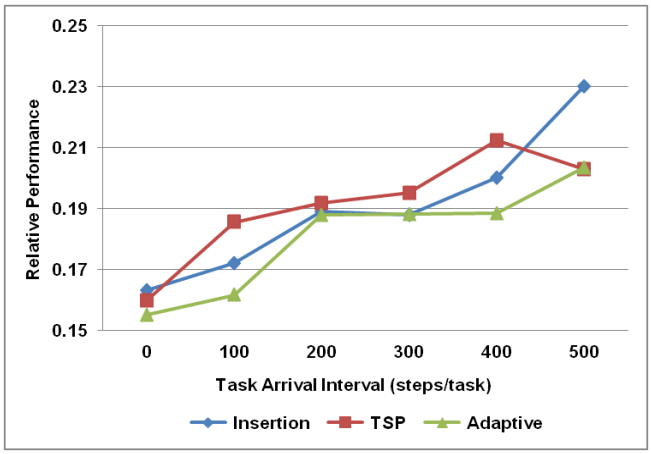

Figure 4. Relative performance of different scheduling policies compared to FCFS, with fixed task count and changing task arrival interval.

5. CONCLUSION AND FUTURE WORK

UAVs will continue to have an important role in current and future military and civilian operations. As multiple UAVs are organized into a swarm, new challenges need to be addressed and new command and control techniques be developed to effectively manage the swarm. To facilitate efficient mission planning for UAV swarms, in this paper we proposed an agent-based control framework. The framework employs a swarm control agent and multiple UAV agents. The SCA assigns ready tasks in the mission to UAVs that have the minimum estimated costs. The UAVs then locally schedule those tasks into their task queues based on different policies. As a proof-of-concept, a multithread simulation testbed is implemented in Java. Experimental results on different local scheduling policies are also discussed.

In the future, we plan to extend our work significantly. First, an expressive mission specification language will be developed so that testbed users can describe mission input more easily and naturally. Secondly, we plan to incorporate more realistic aspects of UAV flight into the testbed, such as collision avoidance and non-static UAV speed. We also plan to add more scenarios from real world swarm operations into the testbed, such as communication lost between a UAV and the SCA, task failure on UAVs and loss of UAVs in a swarm. Thirdly, we are planning to convert the testbed into an open source project so that other researchers in the community can try it out and give us their comments and suggestions.

Acknowledgement

This research was supported in part under a grant from the AFOSR Award # FA9550-11-1-0351. The authors would like to thank Christian Poellabauer, Hongsheng Lu, Rachael Purta and Ryan McCune for their contributions to the project and their suggestions to this paper.

References

[1] Office of the Secretary of Defense, Unmanned Aircraft Systems Roadmap: 2005 - 2030. Technical Report, Department of Defense, 2005.

[2] Library of Congress, Mini, Micro, And Swarming Unmanned Aerial Vehicles: A Baseline Study. Washington, D.C.2006.

[3] Madey, G.R., Blake, M.B., Poellabauer, C., Lu. H., McCune, R.R., and Y. Wei. 2012. "Applying DDDAS Principles to Command, Control and Mission Planning for UAV Swarms," Procedia Computer Science 9 (2012), 1177-1186.

[4] Blake, M.B., and H. Gomaa. 2005. "Agent-oriented Compositional Approaches to Services-based Cross-Organizational Workflow," Decision Support Systems, 40(1), Elsevier, 2005, 31-50.

[5] Wei, Y., and M.B. Blake, 2010. "Service-oriented Computing and Cloud computing: Challenges and Opportunities," IEEE Internet Computing, 14(6), 72-75.

[6] Wei, Y., and M.B. Blake, 2012. "Adaptive Service Workflow Configuration and Agent-based Virtual Resource Management in the Cloud," In Proceedings of IEEE International Conference on Cloud Engineering, San Francisco, CA, March 2013.

[7] Wei, Y., Blake, M.B., and I, Saleh, 2013. " Adaptive Resource Management for Service Workflows in Cloud Environments," In Proceedings of 2nd International Workshop on Workflow Models, Systems, Services and Applications in the Cloud, Boston, MA, May 2013.

[8] Hexmoor, H., Mclaughlan, B., and M. Baker. 2005. "Swarm Control in Unmanned Aerial Vehicles," In Proceedings of International Conference on Artificial Intelligence, CSREA, 2005.

[9] Luke, S., Cioffi-Revilla, C., Panait, L., Sullivan, K., and G. Balan. 2005. "MASON: A Multiagent Simulation Environment," Simulation, 81(7), 517-527.

[10] Rasmussen, S.J., Mitchell, J.W., Chandler, P.R., Schumacher, C.J., and A.L. Smith. 2005. "Introduction to the MultiUAV2 Simulation and Its Application to Cooperative Control Research," In Proceedings of the 2005 American Control Conference, IEEE, 2005, 4490-4501.

[11] Garcia, R., and L. Barnes. 2010. "Multi-UAV Simulator Utilizing X-Plane," In Proceedings of 2nd International Symposium on UAVs, Springer, 2009, 393-406.

[12] Russell, M.A., Lamont, G.B., and K. Melendez. 2005. "On using SPEEDES as a Platform for a Parallel Swarm Simulation," In Proceedings of the 37th Winter Simulation Conference, 1129-1137.

[13] Gaudiano, P., Shargel, B., Bonabeau, E., and B.T. Clough. 2003. "Swarm Intelligence: A New C2 Paradigm with an Application to Control Swarms of UAVs," Icosystem Corp., Cambridge MA.

[14] Varela, G., Caamamo, P., Orjales, F., Deibe, A., Lopez-Pena, F., and R.J. Duro. 2011. "Swarm Intelligence based Approach for Real Time UAV Team Coordination in Search Operations," In Proceedings of 3rd World Congress on Nature and Biologically Inspired Computing, 2011, 365-370.

[15] Lamont, G.B., Slear, J.N., and K. Melendez. 2007. "UAV Swarm Mission Planning and Routing using Multi-Objective Evolutionary Algorithms," In Proceedings of IEEE Symposium on

Computational Intelligence in Multicriteria Decision Making, IEEE, 2007, 10-20.
[16] Vincent, P., and I. Rubin. 2004. "A Framework and Analysis for Cooperative Search using UAV Swarms," In Proceedings of the 2004 ACM Symposium on Applied Computing, ACM, 2004, 79-86.
[17] Dasgupta, P., 2008. "A Multiagent Swarming System for Distributed Automatic Target Recognition using Unmanned Aerial Vehicles," IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans, 38(3), 549-563.
[18] Gaudiano, P., Shargel, B., Bonabeau, E., and B.Clough. 2003. "Control of UAV Swarms: What the Bugs Can Teach Us." In Proceedings of the 2nd AIAA Unmanned Unlimited Systems, Technologies, and Operations Aerospace Conference, San Diego, CA.
[19] Dionne, D., and C.A.Rabbath. 2007. "Multi-UAV Decentralized Task Allocation with Intermittent Communications: The DTC Algorithm." In Proceedings of American Control Conference, IEEE, 2007, 5406-5411.
[20] Dasgupta, P., and M.Hoeing. 2008. "Dynamic Pricing Algorithms for Task Allocation in Multi-Agent Swarms," Massively Multi-Agent Technology, 64-79.
[21] Purta, R., Nagrecha, S., and G.R. Madey, 2013. "Multi-hop Communications in a Swarm of UAVs," In Proceedings of the 2013 Symposium on Agent Directed Simulation (ADS '13/SpringSim 2013). Society for Computer Simulation International, San Diego, CA, 2013.
[22] McCune, R.R., and G.R. Madey, 2013. "Agent-based Simulation of Cooperative Hunting with UAVs," In Proceedings of the 2013 Symposium on Agent Directed Simulation (ADS '13/SpringSim 2013). Society for Computer Simulation International, San Diego, CA, 2013.
[23] Madey, A.G., and G.R., Madey, 2013. "Design and Evaluation of UAV Swarm Command and Control Strategies," In Proceedings of the 2013 Symposium on Agent Directed Simulation (ADS '13/SpringSim 2013). Society for Computer Simulation International, San Diego, CA, 2013.
[24] Wei., Y., Blake, M.B., and G.R. Madey, 2013. "An Operation-time Simulation Framework for UAV Swarm Configuration and Mission Planning," In Proceedings of International Conference on Computational Science, Barcelona, Spain, June 2013.
[25] Sujit, P.B., Kingston, D., and R.Beard. 2007. "Cooperative Forest Fire Monitoring using Multiple UAVs," In Proceedings of 46th IEEE Conference on Decision and Control, IEEE, 2007, 4875-4880.
[26] The JFreeChart library: http://www.jfree.org/jfreechart/ Access December 6, 2012.

Biography

Yi Wei is a Ph.D candidate in the Department of Computer Science and Engineering at the University of Notre Dame. His research areas are agent-based modeling and simulation, service-oriented computing, and distributed computing. He is a student member of ACM and IEEE.

M. Brian Blake is a professor in the Department of Computer Science at the University of Miami. His research areas are agents and workflow, service-oriented computing, and software engineering. He is a senior member of IEEE Computer Society and a distinguished scientist of ACM.

Gregory Madey is a Research Professor in the Department of Computer Science and Engineering at the University of Notre Dame, IN, USA. His research includes topics in bioinformatics, emergency management modeling and simulation, agent-based modeling and simulation, cyber-infrastructure, web portals for scientific collaboration, and scientific databases. He is a member of the ACM, AIS, IEEE Computer Society, Informs, and the Society for Computer Simulation.