Available online at www.sciencedirect.com

SciVerse ScienceDirect

Procedia Computer Science 18 (2013) 1949 – 1958

Procedia Computer Science

2013 International Conference on Computational Science

An Operation-time Simulation Framework for UAV Swarm Configuration and Mission Planning

Yi Wei[a*], M. Brian Blake[a,b], Gregory R. Madey[a]

[a] Department of Computer Science and Engineering, University of Notre Dame, Notre Dame, IN 46556, USA
[b] *Department of Computer Science, University of Miami, Coral Gables, FL 33124, USA*

Abstract

In recent years, Unmanned Aerial Vehicles (UAV), have been increasingly utilized by both military and civilian organizations because they are less expensive, provide greater flexibilities and remove the need for on-board pilot support. Largely due to their utility and increased capabilities, in the near future, swarms of UAVs will replace single UAV use. 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. A centralized-distributed hybrid control framework is proposed for mission assignment and scheduling. The Dynamic Data Driven Application System (DDDAS) principles are applied to the framework so that it can adapt to the changing nature of the environment and the missions. A prototype simulation program is implemented as a proof-of-concept of the framework. Experimentation with the framework suggests the effectiveness of swarm control for several mission planning mechanisms.

Selection and peer review under responsibility of the organizers of the 2013 International Conference on Computational Science

Keywords: DDDAS; UAV Swarm; Mission Planning; Simulation Framework

1. Introduction

Unmanned Aerial Vehicle (UAV), or as it is usually called, drone, 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 remotely in a ground station. The history of using UAVs in warfare can be dated back to World War I [1], but only since the last decade has there been a large scale use of UAV

* Corresponding author. Tel.: +1-312-520-4983 ; fax: +1-574-631-9260
E-mail address: ywei1@nd.edu.

1877-0509
Selection and peer review under responsibility of the organizers of the 2013 International Conference on Computational Science
doi:10.1016/j.procs.2013.05.364

deployments in various military and civilian operations, such as anti-terrorism, remote area surveillance and hazardous environment monitoring.

Compared with traditional piloted airplanes, UAVs are less expensive and can be made into much smaller sizes to increase their flexibilities. Pilot safety is no longer a concern. As their costs 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 and possibly hostile environments [2]. Currently, a UAV is operated by at least one ground operator, sometimes more. This pattern cannot scale as the number of UAVs in a swarm increases rapidly. So, more intelligent command and control approaches and systems are required to let ground pilots "fly the swarm" instead of individual UAVs.

During a swarm's operation, its member UAVs' status, as well as the external environment, change over time. So it is important that the swarm adapts to these changes and adjusts its behaviors accordingly. The Dynamic Data Driven Application System (DDDAS) paradigm, as proposed by Darema [4], is a means to inject runtime data into system models or simulations to achieve more accurate analysis and more precise controls. In this paper, we extend our previous UAV studies [5][6][24][25], leverage intelligent agent and workflow techniques [7], and further apply our related work for configuration and resource allocation in computing environments [8][9][10]. These past studies, in this paper, are applied with DDDAS principles to the swarm mission planning problem and we propose a centralized-distributed framework for mission assignment and scheduling. In the framework, a central controller is responsible for assigning tasks in a mission to different UAVs based on its latest knowledge about the swarm. Upon receiving the task assignment, the UAV tries to schedule the new task into its local task queue in order to minimize total task completion cost. Since the central controller does not have real time information about the swarm, it incorporates the status information sent from the swarm and utilizes this information to update its swarm model for future task assignments. To evaluate the proposed framework, a prototype simulation framework is implemented.

The rest of this paper is organized as follows. In Section 2, related research work in the area of 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 is the descriptions of the proposed mission planning framework. In Section 4, the architecture and implementation of the simulation framework 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 [12] is a general purpose multiagent simulation library that is utilized by our previous work, along with the Matlab based UAV simulator MultiUAV2 [13]. In [11], a swarm simulator is implemented in Java for target searching. Garcia et al. in [14] introduced a multi-UAV simulator implemented on top of a commercial flight simulator called X-Plane. Russell et al. in [15] presented a parallel swarm simulation environment utilizing an existing parallel emulation and simulation tool called SPEEDS. Gaudiano et al. in [16] 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 [17], but in a civilian usage scenario. The problems of path planning and vehicle routing are investigated in [18] using multi-objective evolutionary algorithms. In [19], the cooperative searching problem is discussed for the purpose of detecting moving and evading targets in a hazardous environment. [20] investigates the Automatic Target Recognition (ATR) problem in UAV control and proposed a distributed strategy for UAV swarms. The task allocation problem is discussed in [21] and [22] using different methodologies. There are also applications using a UAV swarm to monitor and detect wild fire hotspots [23].

3. DDDAS driven Swarm Mission Planning

3.1. Definitions

In this subsection, formal definitions of the terms used throughout the rest of this paper are given.

3.1.1. UAV Swarm

A *swarm* consists of a group of UAVs. Let $S = \{v_1, v_2, \ldots, v_n\}$ 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, could be fuel, battery power, etc.

During the swarm's mission, 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, the UAV's maximum capacity and this value together determine how long the UAV can stay in the operation area before having to return to base. This measurement may change overtime.
5. A list of tasks assigned to this UAV.

A swarm's state consists of two parts. One part includes the collection of its UAV's states, the other part 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.

3.1.2. Missions and Tasks

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 the dependencies between different tasks. In D, $\{t_i, t_j\}$ 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 tasks that are in progress, and W is the 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. A DDDAS driven Hybrid 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, M, from the set of tasks to the set of UAVs. 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. But 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.

Let k denote the total number of missions assigned to a swarm, and N is the total number of UAVs in the swarm, the problem can be written as the following:

$$\min \sum c(MS_i),\ 1 \le \mathrm{i} \le k \tag{1}$$

$$f(ui) \ge 0,\ for\ 1 \le i \le n$$

In (1), $c(MS_i)$ is the cost of mission MS_i. $f(u_i)$ is the residual capacity of the *ith* UAV after it completes all assigned tasks. The constraint is that every UAV must have enough capacity to finish all assigned tasks.

To solve this problem, we propose a centralized-distributed hybrid framework which combines global task assignment and local task scheduling. There are two major components in the framework. The first is a *central controller,* and the other is the *UAV swarm*. The central controller in the base station is responsible for assigning tasks in a mission to UAVs, while each UAV that gets a task locally schedules how to finish it.

In order to obtain the latest knowledge about the swarm, the central controller periodically sends status inquiries to all UAVs. Upon receiving such inquiry, a UAV will generate a status message and send it. The message contains the following fields:

1. Current location of the UAV;
2. Residual capacity, in percentage of the UAV's maximum capacity;
3. Calculated capacity consumption rate;
4. Estimated final location after all of its tasks are completed;
5. Estimated final residual capacity after completing all of its tasks.

In addition to this status message, the UAV also sends a task completion message to the central controller every time it finishes a task so that the latter can update the mission information and trigger new task assignment actions.

Figure 1 shows how DDDAS principles are integrated into our control framework. The central controller is the application system as in the DDDAS paradigm, while the simulation swarm is the external data source and different kinds of messages sent from UAVs are the measured data about the real world. Such data is dynamically injected into the running controller to update its knowledge about the swarm and the missions. It then uses this updated knowledge to perform new task assignment or reassignment actions.

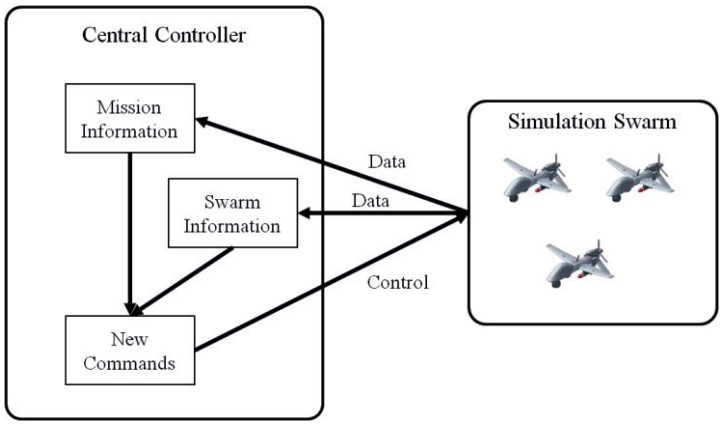

Fig 1. The DDDAS-driven control framework for swarm mission planning

Since tasks in a mission have dependencies among each other, they cannot be all assigned at the same time. So the central controller 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 central controller monitors the status of each mission and assigns tasks to UAVs as they become ready. For each ready task, the central controller selects the assignment target UAV by the following steps:

1. Calculate, for each UAV, the cost of completing t_i. The central controller will use its latest knowledge about the swarm to estimate the cost of finishing t_i. This cost is different for different UAVs, because UAVs are on different locations and may or may not have existing tasks. There are two different situations:
   1.1. 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 capacity;
   1.2. There are scheduled tasks on the UAV already. In this case, the *estimated final location* and *estimated final residual capacity* values will be used to calculate the task cost.
2. The calculated task cost is compared to the UAV's residual capacity. If the former is greater than or equal to the latter, the UAV is selected as a candidate for task assignment;
3. Sort the calculated costs and choose the UAV with the smallest cost value as the candidate for t_i.

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 central controller 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 u_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 u_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* the cost of the new schedule which includes t_m. Let function $D(l_i, l_j)$ denote the Euclidean 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:

$$ns = min(os + D(l_k, l_m) + C(t_m), min(os - D(l_i, l_{i+1}) + D(l_i, l_m) + D(l_m, l_{i+1}) + C(t_m)), 0 \le i < k \quad (2)$$

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. *Traveling Salesman Problem* (TSP): the scheduling problem can also be modeled 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 Euclidean 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 neighbor. 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 central controller. After receiving such request, the central controller will reassign those tasks to other UAVs.

4. Simulation Framework

A simulation framework is built as a proof-of-concept of the proposed DDDAS driven mission planning framework. The framework is implemented as a multithread Java program, and both the central controller and all UAVs are represented by threads. Communications between the central controller and UAVs are implemented as 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 contains descriptions of all messages used in the framework. We can see that a participant of the framework, either a UAV or the controller, can become both message sender and receiver. Each participant also has two message queues, one for incoming messages and the other for outgoing ones.

Each UAV is under the governance of a control loop. The loop body has the following steps:

1. Retrieve messages from the inbound message queue;
2. Process received messages based on their types;
3. If there are tasks scheduled on the UAV, test the feasibility of the schedule;
4. Send messages in the outbound message queue to their receivers;
5. Update the internal state of the UAV, including its location, heading, and residual capacity.

Steps 2 and 3 may trigger the generation of outbound messages. At the beginning of a simulation run, each UAV is given a list of randomly generated default waypoints to follow, thus forming a default trajectory. If a new task is assigned and scheduled on the UAV, it will leave the default trajectory for the task location. After all scheduled tasks are completed, the UAV will choose the nearest default waypoint and return to that location to resume its default trajectory.

Similar to the UAV, the central controller also has a control loop with the following steps:

1. Receive messages from inbound message queue;
2. Process received messages based on their types;
3. Send out all outbound messages;
4. Send status inquiry messages to all UAVs.

The central controller incorporates the latest information contained in the received messages into its knowledge about the swarm and all missions. When new mission arrives, the central controller uses the mission planning process discussed in Section 3.2 to select a best UAV candidate.

Figure 2 shows the GUI of the framework, implemented using Java Swing. The left part of the GUI has the mission output area on top and the control buttons at the bottom. The framework 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. In Figure 2, two missions, M1 and M2, with a total of 13 tasks, are given to the central controller for planning and execution. And task T1 in mission M1 has been completed, task T2 is being executed by one of the UAVs, while task T3 has been scheduled on a UAV and is waiting to be executed.

At the bottom left of the GUI are the control buttons, provided for the convenience of framework users to allow them to take a closer look at the swarm and the missions. 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 utilized to generate simulation report charts, as shown in Figure 3. The left part of Figure 3 is a pie chart showing how many tasks are completed by each UAV in the swarm, while the bar chart on the right part shows the total traveling distance of each UAV. 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 framework visualizes in this area all UAVs in the swarm, 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 colors of UAVs, tasks and waypoints are randomly selected during simulation to differentiate each other. 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 2. Communication messages between different participants of the framework

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

Table 3. Task colors and their indicated task status

Task Color	Task Status
Black	Task hasn't been assigned yet because its predecessors have not completed.
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.

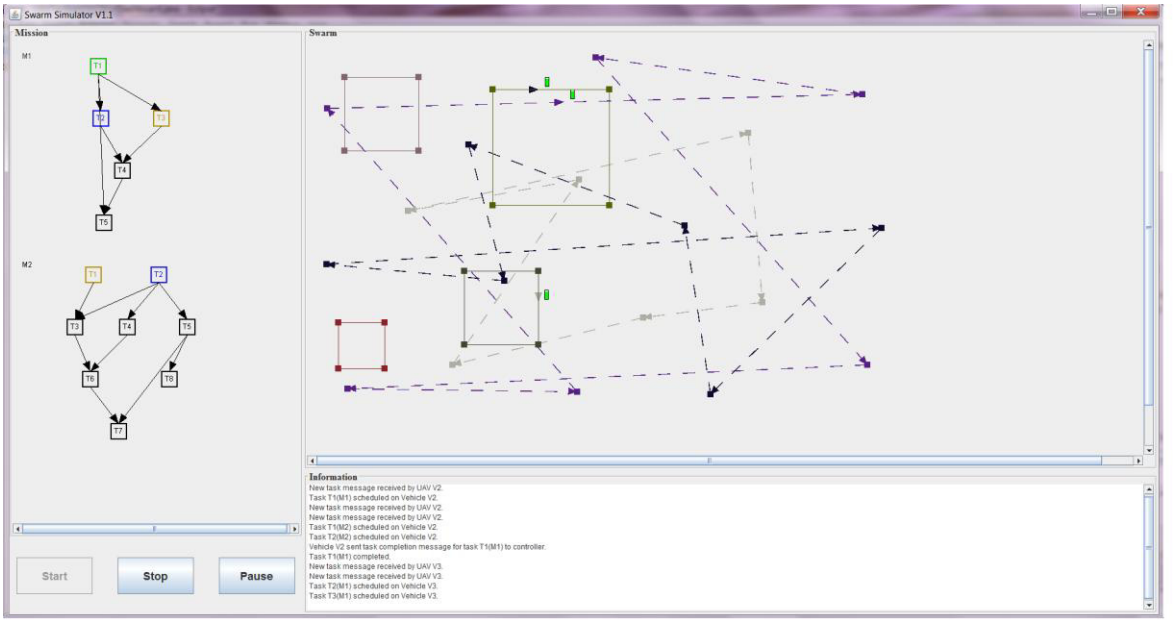

Fig. 2 Screenshot of the simulation framework in execution with 3 UAVs and 2 missions

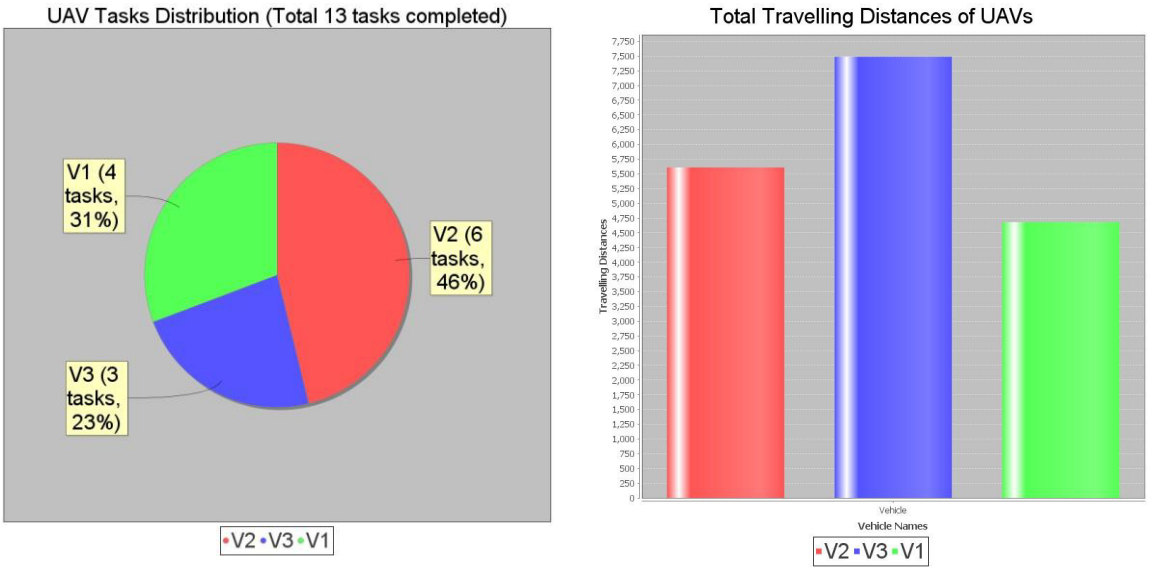

Fig. 3. A pie chart (left) and a bar chart (right) generated by the framework

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 a centralized-distributed hybrid framework. The framework applies DDDAS principles to dynamically incorporate latest swarm information into the mission planning process. A central controller 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 framework is implemented in Java.

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

Acknowledgements

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] History of Unmanned Aerial Vehicles: http://en.wikipedia.org/wiki/History_of_unmanned_aerial_vehicles. Accessed December 6, 2012.

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

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

[4] Darema, F., 2004. Dynamic Data Driven Applications Systems: A New Paradigm for Application Simulations and Measurements, *Computational Science-ICCS*, Springer, 2004, p. 662-669.

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

[6] Wei, Y., Blake, M.B., Madey, G.R., 2013, Agent-based Simulation for UAV Swarm Mission Planning and Execution, *Agent-Directed Simulation Symposium*, SCS (2013).

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

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

[9] Wei. Y., Blake, M.B., 2012. Adaptive Service Workflow Configuration and Agent-based Virtual Resource Management in the Cloud, *IEEE International Conference on Cloud Engineering*, IEEE (2013)

[10] Wei. Y., Blake, M.B., Saleh, I., 2013, Adaptive Resource Management for Service Workflows in Cloud Environments, 2nd International Workshop on Workflow Models, Systems, Services and Applications in the Cloud (CloudFlow 2013), IEEE (2013).

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

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

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

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

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

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

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

[18] Lamont, G.B., Slear, J.N., Melendez, K., 2007. UAV Swarm Mission Planning and Routing using Multi-Objective Evolutionary Algorithms, *IEEE Symposium on Computational Intelligence in Multicriteria Decision Making*, IEEE, 2007, p. 10-20.

[19] Vincent, P., Rubin, I., 2004. A Framework and Analysis for Cooperative Search using UAV Swarms, *Proceedings of the 2004 ACM Symposium on Applied Computing*, ACM, 2004, p. 79-86.

[20] 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), p. 549-563.

[21] Dionne, D., Rabbath, C.A., 2007. Multi-UAV Decentralized Task Allocation with Intermittent Communications: The DTC Algorithm, *Proceedings of American Control Conference*, IEEE, 2007, p. 5406-5411.

[22] Dasgupta, P., Hoeing, M., 2008. Dynamic Pricing Algorithms for Task Allocation in Multi-Agent Swarms, *Massively Multi-Agent Technology*, p. 64-79.

[23] Sujit, P.B., Kingston, D., Beard, R., 2007. Cooperative Forest Fire Monitoring using Multiple UAVs, *Proceedings of 46th IEEE Conference on Decision and Control*, IEEE 2007 p. 4875-4880.

[24] Purta, R., Dobski, M., Jaworski, A, Madey, G., 2013. A Testbed for Investigating the UAV Swarm Command and Control Problem Using DDDAS, *Procedia Computer Science* 10, 2013.

[25] McCune, R. R., Madey, G. R., 2013, Agent-Based Simulation of Cooperative Hunting with UAVs, *Procedia Computer Science* 10, 2013.

[26] The JFreeChart library: http://www.jfree.org/jfreechart/ Access December 6, 2012.