Available online at www.sciencedirect.com

SciVerse ScienceDirect

Procedia Computer Science 18 (2013) 2537 – 2544

Procedia Computer Science

International Conference on Computational Science, ICCS 2013

Swarm Control of UAVs for Cooperative Hunting with DDDAS

R. Ryan McCune[a], Gregory R. Madey[a]

[a]*Department of Computer Science and Engineering, University of Notre Dame, Notre Dame, IN 46556, USA*

Abstract

Swarm control is a problem of increasing importance with technological advancements. Recently, governments have begun employing UAVs for reconnaissance, including swarms of drones searching for evasive targets. An agent-based simulation for dynamic cooperative cleaning is augmented with additional behaviors and implemented into a Dynamic Data-Driven Application System (DDDAS) framework for dynamic swarm control.

Keywords: DDDAS, Swarm Intelligence, Agent-Based Simulation;

1. Introduction

Swarm intelligence has received much research attention in recent years. Swarm intelligent systems generally exhibit decentralized control achieved through simple agent behaviors and interactions developing a self-organization that is considered an emergence of order from the system. Together the system is greater than the sum of its parts, capable of solving complex problems that no agent could accomplish singularly. Common examples may be observed in nature, like birds adjusting flight parameters relative to neighbors to yield flock formations, or ants depositing pheromone trails that facilitate efficient discovery of optimal food routes.

1.1. Swarm Applications

Governments are investigating the development of swarms of unmanned aerial vehicles (UAVs), or drones, for search and reconnaissance, in instances such as lost hikers or forest fires. Cooperating UAVs can complete more elaborate tasks, but at the expense of increased complexity owed to additional communication and computational demands [1]. Swarm intelligent UAVs offer an attractive solution of structures that are robust, adaptive, scalable, decentralized, and require little communication [2]. Much research has characterized frameworks for UAV swarms but few behaviors or solvable problems have been outlined.

The dynamic cooperative cleaners problem characterizes the task of agents cleaning a grid containing a number of dirty tiles, with dirtiness spreading to neighboring tiles at an established rate. The cooperative hunters problem describes agents searching for intelligently evasive targets. Research in [3] demonstrates that any cleaning protocol may be used as a hunting protocol. Two decentralized cooperative search algorithms, the Parallel Path Search [4] and the SWEEP Protocol [5], exhibit swarm intelligence. Parallel Path Search, also known as the UCLA algorithm, organizes agents side-by-side to form a line to fly back-and-forth parallel to the boundary region, with a certain

*Corresponding author. Tel.: +1-574-631-7596.
E-mail address: rmccune@nd.edu.

Selection and peer review under responsibility of the organizers of the 2013 International Conference on Computational Science
doi:10.1016/j.procs.2013.05.436

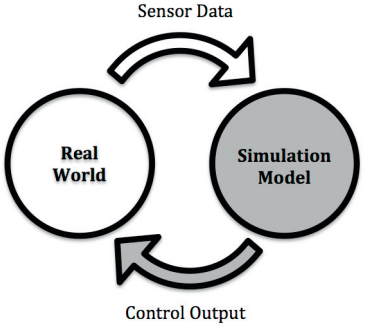
amount of overlap to account for dirty tile expansion. In contrast, the SWEEP Protocol, or the HUNT-II algorithm, arranges agents around the perimeter of the unsearched, or ”dirty”, space, and repeatedly cleans the perimeter so long as the dirty space does not become disconnected. The two methods possess relative strengths and weaknesses, namely, search with the UCLA algorithm is restricted to predetermined rectangles, while the HUNT-II algorithm can clean unknown (non-rectangular) areas but requires knowledge of the entire grid and increased computation to determine whether or not a cleaned tile will disconnect the dirty region.

By adding a third behavior, that of a sentry UAV, complex maps can be partitioned into sub-regions, where each sub-region can be cleaned by a selected number of UAVs and cleaning algorithm. Consider a stationary UAV capable of powering a stronger, larger sensor because of power conserved by reduced mobility. Placing these sentries at selected points on a map, e.g. physical bottlenecks like a tunnel or mountain pass, divides the map into more manageable sub-components, as enemies can no longer expand into neighboring territories unidentified. Partitioning the map into subcomponents leads to an optimization problem of how to best partition the map based on available UAVs, search time, and costs of communication, coordination, and computation.

1.2. DDDAS

Dynamic Data Driven Application Systems (DDDAS) describes a paradigm where a system incorporates simulated data into the real-time decision process while retaining the ability to dynamically manage sensors to refine measurements. New sensor data is assimilated into the running simulation, enabling a synergistic feedback and control-loop between the real-world and the simulation model, portrayed in Figure 1. The synergy is realized through a simulation capable of modeling complex, non-linear dynamics with faster-than-realtime results. Moreover, sensor controls continually drive the measurement process in order to recalibrate the simulation and provide precise and meaningful results.

Fig. 1. DDDAS controls through simulation



The DDDAS paradigm has been applied to a variety of problems, from managing forest fires [6] to crisis management [7] to oceanography [8] but is not without complications. Implementing DDDAS poses several challenges, including the conceptual design of the system and simulation pairing, the assimilation of real-time data into a running application, and accounting for error in a feedback system. Possible approaches are outlined in the following sections.

1.2.1. Simulation Design

Implementing the DDDAS paradigm requires a conceptual understanding of the task at hand. The type of simulation to incorporate into the system depends on the problem being addressed, namely, what the designer hopes to achieve. Previous DDDAS implementations demonstrate two main approaches to the simulation architecture. The first approach is an application that includes a continually updated model, where sensor measurements fed into the model and the system control is recalibrated to minimize a function, e.g. error. This continous control simulation is exemplified in [9] where surgical parameters are continually recalibrated to reduce error, and in [10] where a soil system model is repeatedly updated with real-time data to predict water infiltration rates.

In contrast to continuous recalibration, a system may be designed with a simulation that only runs after detection of a particular event or anomoly. Streaming data is continuously queried and tested, and once an event

is detected, executes a new simulation. In [11] a weather system continually runs a cluster algorithm to identify extreme weather events, and in [7] a DDDAS implementation tries to predict a crisis only after anomalous behavior is detected. The type of simulation to incorporate into a DDDAS implementation must properly reflect the purpose of the system.

1.2.2. Data Assimilation

Real-time data assimilated into a dynamic simulation is a distinctive feature of DDDAS with several considerations. If the simulation cannot continuously incorporate new data, then an interval at which to begin new execution must be established that would execute based on new and possibly previous or even predicted data. Should a time interval be utilized, then the simulation output and resulting control decisions must be capable of execution in less time than the interval. A control model repeatedly predicting a certain number of time steps into the future is considered a receding horizon control [12].

The simulation drives system measurement but as sensors are prone to pertubations, a feed-back loop like DDDAS is prone to error. A DDDAS implementation must account for possible errors in the system. The Kalman Filter [13], and more recently the Ensemble Kalman Filter [14], is a statistical least-squares method to account for noise and missing data in a time-series. The method has been subject to extensive study and testing, and found numerous applications in autonomous or assisted navigation and weather modeling. The technique has also been applied to several DDDAS approaches [6].

1.3. Our DDDAS Implementation for Cooperative Cleaning Mission Control

Any mission incorporating UAVs is inherently dynamic. Smaller UAVs are more vulnerable, prone to failure, and provide less reliable sensor data, making it necessary for mission designers to account for dynamism into planning. For optimal sentry and dynamic cooperative cleaning algorithm assignment, we employ a Dynamic, Data-Driven Application System (DDDAS) incorporating agent-based simulation.

DDDAS entails a system that incorporates dynamic data flow into a running simulation, and conversely, may employ the simulation to adaptively control a real-time application [15, 16, 6]. The agent-based model, developed using the MASON framework, is capable of receiving sensor data and flight status data from the UAV swarm and updates models to predict future performance, e.g. if a drone fails, by utilizing the DDDAS framework, a swarm may be reassigned new search and sentry protocols to best meet mission objectives. An agent-based model is presented for mission control.

Agent-based simulations are attractive models for implementing a DDDAS framework because of the nonlinear dynamics captured by the simulation. Many phenomena best modeled by agent-based simulations are inherently difficult to model by other methods. Phenomena modeled usingagent-based simulation are generally more complex, non-deterministic, and difficult to portray through equations [17].

For the cooperative hunters problem, an agent-based simulation is employed to model a UAV deployment scenario and determine whether or not a topology can be cleaned. The DDDAS framework is incorporated by accounting for the dynamism of a mission where a UAV may fail at any time. When the system detects a UAV has gone down, a simulation is driven by the event and re-simulates the scenario with missing UAV removed. Depending on the stage of the deployment, as well as the map and deployment parameters, the mission may still complete without the downed drone. If not, the system notifies the mission control of the problem status and prompts for drone reassignment or a new control control parameter for the entire swarm.

2. Related Work

Much research effort has been put forth to apply emergent properties of swarm intelligence to other problems. The Particle Swarm Optimization [18] was one of the first successful applications, utilizing agents traveling through a search space based on local results to efficiently find global optima. Ant Colony Optimization [19] also successfully applies principles of swarm intelligence found in nature to optimization problems in machine learning. Methods to control swarms of UAV have also been investigated. Agent-based models have explored UAV control using digital pheromones [1] and emergent phenomena [20]. Multi-agent properties such as bidding are explored within the UAV context [21]. UAV behaviors for a taxonomy of target searching is presented [22]

as well as approaches for coverage in wireless sensor networks [23]. A framework for designing UAV mobility models for reconnaissance is addressed [24]. Similarly, distributed autonomous robot behaviors are presented for mapping.

The framework presented for distributed swarms of cooperative hunters extends the two cleaning algorithms compared by Altshuler et al [25], [26], [3], [27]. The original UCLA algorithm was presented in [4], the Hunt-II algorithm in [5], the two were compared in [25], bounds and analysis for the two approaches are explored in [3] and [27]. The proof that solutions to dynamic cooperative cleaning also solve cooperative hunters is given in [25]. Various lower level technicalities of the HUNT-II protocol were outlined in [26]. Previous work in [28], [29], and [30] explores DDDAS for UAV mission planning.

3. Cooperative Cleaning

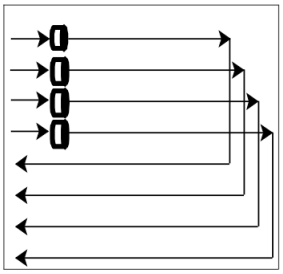
The Dynamic Cooperative Cleaners problem characterizes the task of agents cleaning a grid containing a number of "dirty" tiles, with dirtiness spreading to neighboring tiles at an established rate. When an agent visits a dirty tile, the tile becomes clean in a single timestamp, and the agent moves on to an neighboring tile on the next time stamp. Any tile may become dirty again over time if a neighboring tile is also dirty. The Cooperative Hunters Problem describes agents searching for intelligently evasive targets, where agents must search an area in such a manner so as to be certain where targets may no longer exist. Research in [3] demonstrates that any cleaning protocol may be used as a hunting protocol.

Governments and other public services are investigating how swarms of UAVs may be utilized for complex probles, such as search and reconnaissance under adverse conditions. Considering that some entities in a search region may avoid detection, the task of the UAV swarm may be considered a Cooperative Hunters problem [4]. The UAV swarm attempts to guarantee that, for a given area, all evasive targets are identified by a certain time, and that the time is minimal. Two swarm-intelligent algorithms are presented.

3.1. The Parallel Path Planning Algorithm

The Parallel Path Algorithm, an original implementation of swarm search, is depicted as a homogenous group of UAVs with constant velocity and equivalent sensing capabilities [4]. The search region is a rectangle with dimensions known advance so UAVs can properly account for dirt rate of expansion and overlap search regions if necessary when changing direction. Figure 2 from [4] depicts a flight path for vertically aligned UAVs. In instances of a dynamic dirty region that spreads at a predetermined rate, pictured in Figure 3 and described in [27], overlapping paths are necessary for UAVs to sufficiently clean a dynamic dirty region. Refer to [4] for a formal problem definition.

Fig. 2. Parallel Path line formation where agents sweep back and forth in a line [4], not unlike a group of lawnmowers



3.2. The SWEEP Protocol

The SWEEP Protocol applies a general rule from computer graphics by preserving the connectivity of the dirty region by preventing agents from cleaning critical points tiles which would disconnect the graph if otherwise cleaned [3]. Agents are placed along the perimeter of the dirty region and all move in the same direction along the border. Agents can detect whether the given location is dirty or not, and possess a limited vision, capable of

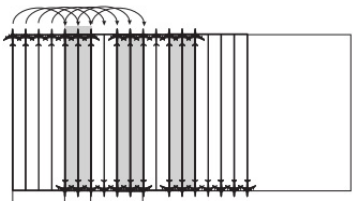
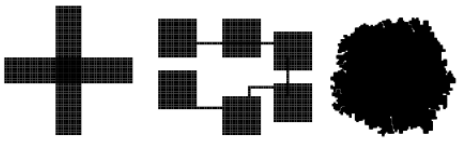


Fig. 3. While agent sweep back and forth in a line, the dynamic dirty area continues to expand, so agents must overlap as shown by the shaded area [27]

Fig. 4. The SWEEP Protocol, adapted from computer graphics, is capable of cleaning any connected area by placing agents along the perimeter and moving inward, cleaning as long as the tile does not disconnect the remaining area [25]



determining the dirtiness of four von Neumann neighbors. The SWEEP Protocol is versatile, requiring no previous knowledge of the topology to ensure cleanliness, permitting the cleaning of complex shapes [25]. However, determining grid connectivity for every agent on every move can become computationally expensive. Moreover, interesting problems arise for directing a simple agent when regions become complex after rounds of cleaning and expanding. For example, rings may form which delay agent cleaning, or, as depicted in Figure 5 [3] the Swiss Cheese Trap may occur when an interior hole confuses an agent. Refer to [3] for the protocol introduction and [26] for the most detailed technical discussion.

3.3. The Sentry

An addition to the two previously established cooperative cleaning algorithms, a third behavior of a "sentry" is added. A sentry is a UAV that arrives to a single provided location and loiters, cleaning (or monitoring) the local tiles within the cleaning radius of a UAV. Agents in the two cleaning algorithms are only capable of cleaning the tile that the agent currently inhibits. However, due to power conserved by remaining stationary, sentry UAVs may sense and clean neighboring tiles. In current simulations, sentry agents are capable of cleaning the current tile location and all 8-neighbor tiles. The added capability of a sentry adds dramatically new capabilities for swarm cleaning. The sentry exemplifies methods to partition the search space, allowing different cleaning algorithms to be assigned to sub-regions. Currently the sentry is centrally assigned.

4. Agent-Based Simulation

Three agent-based simulations were developed to model solutions to the cooperative hunters problem. All simulations were developed with the MASON Framework, an agent-based simulation framework from [31]. The simulations allow users to determine whether or not a swarm of UAVs can clean a given region under a given set of parameters, that is, guarantee all mobile entities have been identified within the given search map.

The given simulation tests a given solution entered by the user. The simulation is integrated with a DDDAS framework at runtime by monitoring the UAVs during deployment. If a UAV fails, the system detects the failure and the simulation is re-run with the missing UAV. Depending on the stage of deployment and UAVs remaining, the simulation will determine whether or not the cleaning can be completed. If not, mission control is prompted for agent reassignment.

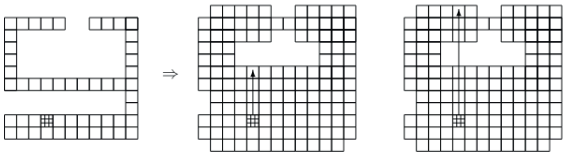
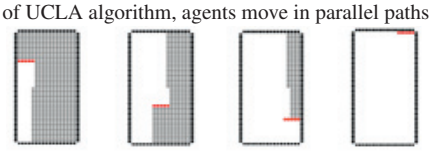


Fig. 5. The SWEEP Protocol is more flexible than the Parallel Path algorithm, but is more complex because agents must know the entire map, and in some instances, make difficult and computationally expensive moves [3]

4.1. Streakers

The first solution implemented was the UCLA algorithm. Topology must be rectangular, and walls are generated along the perimeter so the contaminant may not spread outside the original boundaries. The user inputs the number of agents via the console portrayed in Figure 8, as well as agent velocity, and the contaminant spread rate. Figure 6 displays a set of vertical streakers.

Fig. 6. Vertical implementation of UCLA algorithm, agents move in parallel paths back and forth to clean the map

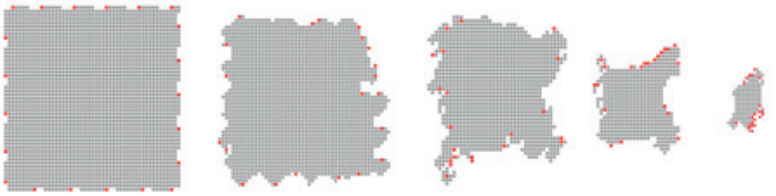


4.2. Sweepers

The SWEEP Protocol was also implemented. For a given map, the user specifies the number of agents, agent speed, and contaminant spread rate via the console. Agents are automatically placed at equal distances around the perimeter of the contaminant. Walls are not required for Sweepers like required by Streakers.

Second, the keystone of Sweeper behavior is the rule that no tile be cleaned that would disconnect the contaminant region. This requires the connectivity of the entire contaminant be tested on every step of every agent, which quickly becomes computationally expensive. One solution was to implement agent "vision" where an agent can see in all directions from a current location (including diagonals) for a given radius. When an agent tests if a cleaning a tile would disconnect the region, the agent first tests if cleaning the tile would disconnect the region within the line-of-sight of an agent. If so, then the entire topology is tested. Otherwise the tile is cleaned. A simulation of Sweepers cleaning a square is depicted in Figure 7. The simulation is useful to visualize how a region is cleaned.

Fig. 7. The SWEEP Protocol cleaning a 48x48 square, where all agents begin equally spaced along the perimeter and move clockwise in unison, gradually moving inward, and adapting to the dynamic growth of the dirty tiles

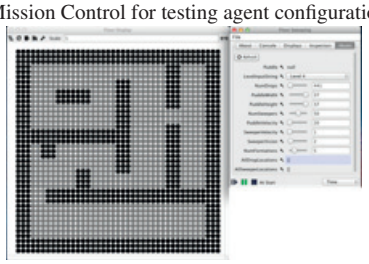


4.3. Sentry Partitioning Simulation

A third simulation is under development to incorporate both Streaker and Sweeper algorithms by allowing a topology to be partitioned with Sentries. Through mouse clicks on the console, a user would select locations on the map for the sentries, and then assign a cleaning algorithm to each sub-region. This will be useful for simulating different scenarios and partition assignments according to costs associated with implementing each algorithm.

Figure 8 depicts the console with a newly instantiated map, and input console for mission parameters including number of agents, velocity, spread rate, etc.

Fig. 8. DDDAS Mission Control for testing agent configurations on a new map

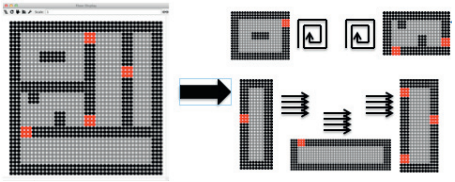


The mission controller would review the map and select points to add Sentry agents. In Figure 9, the controller has intelligently assigned Sentries to map locations that partition the map into less complex sub-regions. The controller is aware that in this mission, Sentries have a cleaning radius of 1, thus cleaning a 3x3 square.

Sentries can be assigned anywhere, but a given mission may have a restricted number of UAVs and the goal is to minimize total cleaning time and/or other metrics associated with the cost of cleaning, e.g. communication or computation. In the demonstrated example, four Sentries were assigned that partitioned the map into five sub-regions, shown in Figure 9 with their respective algorithm assignments.

The end result will allow a controller to assign sentries to locations on a map, and test different algorithms and assignments within mission context.

Fig. 9. Sub-region Partitioning tests if the configuration can be cleaned, then dynamically recalculates if the map can be cleaned when an agent fails, implementing an event-driven DDDAS framework



5. Conclusions and Future Work

The addition of sentry UAVs facilitates spatial partitioning of a map into sub-regions, creating an optimization problem of how to best partition the map into sub-regions considering trade-offs between two swarm cleaning algorithms. The current simulation is useful to test different scenarios depending on mission contexts and determine whether or not a region can be cleaned. The simulation was integrated into a DDDAS framework to develop a dynamic system capable of adapting to real-time mission developments, like UAV failure. The simulation also exposes different strategies and trade-offs, and reveals computational considerations.

Future work includes developing a model for inter-agent communication and swarm computation. With such models, performance of a particular partitioning and assignment can be formally evaluated and compared, in addition to being tested for cleaning completeness. One observation is that Streakers can merely follow a leader that does not require much orientation, while comparatively Sweepers must share information regarding the map and avoid collisions, in addition to already complicated behavioral programming. With a communication and computational model, a formal optimization problem can be constructed.

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 Yi Wei for their contributions to the project and their suggestions to this paper.

References

[1] P. Gaudiano, B. Shargel, E. Bonabeau, B. Clough, Swarm intelligence: A new C2 paradigm with an application to control swarms of uavs, Tech. rep., DTIC Document (2003).

[2] E. Şahin, Swarm robotics: From sources of inspiration to domains of application, Swarm Robotics (2005) 10–20.

[3] Y. Altshuler, A. Bruckstein, I. Wagner, Swarm robotics for a dynamic cleaning problem, in: Swarm Intelligence Symposium, 2005. SIS 2005. Proceedings 2005 IEEE, IEEE, 2005, pp. 209–216.

[4] P. Vincent, I. Rubin, A framework and analysis for cooperative search using uav swarms, in: Proceedings of the 2004 ACM symposium on Applied computing, ACM, 2004, pp. 79–86.

[5] I. A. Wagner, Y. Altshuler, V. Yanovski, A. M. Bruckstein, Cooperative cleaners: A study in ant robotics, The International Journal of Robotics Research 27 (1) (2008) 127–151.

[6] C. Douglas, J. Beezley, J. Coen, D. Li, W. Li, A. Mandel, J. Mandel, G. Qin, A. Vodacek, Demonstrating the validity of a wildfire dddas, Computational Science–ICCS 2006 (2006) 522–529.

[7] T. Schoenharl, G. Madey, Design and implementation of an agent-based simulation for emergency response and crisis management, Journal of Algorithms & Computational Technology 5 (4) (2011) 601–622.

[8] N. Patrikalakis, J. McCarthy, A. Robinson, H. Schmidt, C. Evangelinos, P. Haley, S. Lalis, P. Lermustaux, R. Tian, W. Leslie, et al., Towards a dynamic data driven system for rapid adaptive interdisciplinary ocean forecasting, Dynamic Data-Driven Application Systems. Kluwer Academic Publishers, Amsterdam.

[9] J. Oden, K. Diller, C. Bajaj, J. Browne, J. Hazle, I. Babuška, J. Bass, L. Demkowicz, Y. Feng, D. Fuentes, et al., Development of a computational paradigm for laser treatment of cancer, Computational Science–ICCS 2006 (2006) 530–537.

[10] H. Yu, C. C. Douglas, F. L. Ogden, A new application of dynamic data driven system in the talbot-ogden model for groundwater infiltration, Procedia Computer Science 9 (0) (2012) 1073 – 1080, Proceedings of the International Conference on Computational Science, ICCS 2012. doi:10.1016/j.procs.2012.04.116.
URL http://www.sciencedirect.com/science/article/pii/S1877050912002372

[11] L. Ramakrishnan, Y. Simmhan, B. Plale, Realization of dynamically adaptive weather analysis and forecasting in lead: Four years down the road, Computational Science–ICCS 2007 (2007) 1122–1129.

[12] D. Q. Mayne, H. Michalska, Receding horizon control of nonlinear systems, Automatic Control, IEEE Transactions on 35 (7) (1990) 814–824.

[13] G. Welch, G. Bishop, An introduction to the kalman filter (1995). Proc of SIGGRAPH, Course 8. 27599-3715.

[14] P. L. Houtekamer, H. L. Mitchell, Data assimilation using an ensemble kalman filter technique, Monthly Weather Review 126 (3) (1998) 796–811.

[15] F. Darema, Dynamic data driven applications systems: A new paradigm for application simulations and measurements, Computational Science-ICCS 2004 (2004) 662–669.

[16] F. Darema, Dddas computational model and environments, Journal of Algorithms & Computational Technology 5 (4) (2011) 545–560.

[17] H. Van Dyke Parunak, R. Savit, R. Riolo, Agent-based modeling vs. equation-based modeling: A case study and users guide, in: Multi-Agent Systems and Agent-Based Simulation, Springer, 1998, pp. 277–283.

[18] J. Kennedy, R. Eberhart, Particle swarm optimization, in: Neural Networks, 1995. Proceedings., IEEE International Conference on, Vol. 4, IEEE, 1995, pp. 1942–1948.

[19] M. Dorigo, M. Birattari, T. Stutzle, Ant colony optimization, Computational Intelligence Magazine, IEEE 1 (4) (2006) 28–39.

[20] E. Bonabeau, C. Hunt, P. Gaudiano, Agent-based modeling for testing and designing novel decentralized command and control system paradigms, Tech. rep., DTIC Document (2003).

[21] H. H. Brian, B. Mclaughlan, M. Baker, Swarm control in unmanned aerial vehicles, in: In Proceedings of International Conference on Artificial Intelligence (IC-AI), CSREA, Press, 2005.

[22] P. Gaudiano, E. Bonabeau, B. Shargel, Evolving behaviors for a swarm of unmanned air vehicles, in: Swarm Intelligence Symposium, 2005. SIS 2005. Proceedings 2005 IEEE, IEEE, 2005, pp. 317–324.

[23] J. Cortes, S. Martinez, T. Karatas, F. Bullo, Coverage control for mobile sensing networks, Robotics and Automation, IEEE Transactions on 20 (2) (2004) 243–255.

[24] E. Kuiper, S. Nadjm-Tehrani, Mobility models for uav group reconnaissance applications, in: Wireless and Mobile Communications, 2006. ICWMC'06. International Conference on, IEEE, 2006, pp. 33–33.

[25] Y. Altshuler, V. Yanovsky, I. Wagner, A. Bruckstein, The cooperative hunters-efficient cooperative search for smart targets using UAV swarms, in: Second International Conference on Informatics in Control, Automation and Robotics (ICINCO), the First International Workshop on Multi-Agent Robotic Systems (MARS), 2005, pp. 165–170.

[26] Y. Altshuler, V. Yanovski, Dynamic cooperative cleanersvarious remarks, Tech. rep., Technical report (2005).

[27] Y. Altshuler, V. Yanovsky, I. Wagner, A. Bruckstein, Efficient cooperative search of smart targets using UAV swarms, Robotica 26 (4) (2008) 551–557.

[28] G. Madey, M. Blake, C. Poellabauer, H. Lu, R. McCune, Y. Wei, Applying DDDAS principles to command, control and mission planning for UAV swarms, Procedia Computer Science 9 (2012) 1177–1186.

[29] Purta, R., Dobski, M., Jaworski, A., Madey, G., A testbed for investigating the UAV swarm command and control problem using DDDAS, in: Procedia Computer Science 10, 2013.

[30] Wei, Y., Blake, M. B., Madey, G., An operation-time simulation framework for UAV swarm configuration and mission planning, in: Procedia Computer Science 10, 2013.

[31] S. Luke, C. Cioffi-Revilla, L. Panait, K. Sullivan, G. Balan, Mason: A multiagent simulation environment, Simulation 81 (7) (2005) 517–527.