(Preprint) AAS 20-024

DECENTRALIZED SPACECRAFT SWARMS FOR INSPECTION OF LARGE SPACE STRUCTURES

Byong Kwon,[*] Jekan Thangavelautham[†]

The emergence of increasingly sophisticated and modular small satellites is expected to enable in-space assembly of large space observatories, space infrastructure, such as propellant depots and communication relays, and larger modular interplanetary spacecraft. The key is the modular assemble of these space architectures that enables quick assembly of more capable structure and spacecraft with longer range to reach unexplored planetoids, moons, and asteroids in the outer solar system. A key task to assembling small modular structures into a larger structure is the need for careful verification to ensure all the pieces are locked in place. Attempts to minimize or eliminate the use of human astronauts for such tasks would be a major technological achievement and welcome simplification of the overall complexity of the system. In this paper, we present a neural network robotic controller, the Artificial Neural Tissue (ANT), to perform decentralized control of multiple robots for optimal area coverage of large structures. With this robotic controller, there is no supervisor or hierarchy among the robots. In computer simulations, robots can achieve near-optimal parallelism, where increasing the number of robots, *n*, allows the task to be completed in T_1/n time, where T_1 is the time for one robot to complete the entire task. The robotic controllers are evolved using Darwinian methods in simulation. The fittest controllers can then be tested in high-fidelity simulations or on robotic hardware. To date, our simulation results show the controller enabling multiple robots to self-assign different regions for different robots and thus minimizing covering the same area twice by a single, or multiple robots. The simulations have been extended to various shape primitives including rectangular, square, circular and triangular areas. We find the controllers being able to repeatedly find optimal or near-optimal solutions without requiring human supervision. In fact, some of the solutions could be considered human competitive as they match or exceed human capabilities in solving the problem. Our next steps are to demonstrate the controllers using high-fidelity dynamics simulators, followed by demonstrations on robotic hardware in laboratory.

INTRODUCTION

In-space assembly of small modular structures into larger structures, such as space observatories, propellant depots, communication relays, and larger modular interplanetary spacecraft is an important capability for future space missions. A key task to enable in-space assembly is careful inspections of large structures to ensure that the modular components are connected and locked properly, in place. Given the overall complexity of constructing such large structures, automated

[*] Postdoctoral Research Fellow, Aerospace & Mechanical Engineering, University of Arizona, Tucson, AZ 85721.
[†] Assistant Professor, Aerospace & Mechanical Engineering, University of Arizona, Tucson, AZ 85721.

inspections by a robot or spacecraft swarms will simplify construction and allow mission planners to re-allocate human astronauts or tele-operators to critical tasks that cannot be automated.

In this paper, we present the Artificial Neural Tissue (ANT), a neural network robotic controller for optimal area coverage of large structures by a decentralized, multi-agent swarm.[1,2] Specifically, we present computer simulations, where an ANT multi-agent swarm completely, or near completely, covers 2-dimensional (2D), basic geometric, open grid areas in linear or quasi-linear time, where time complexity is measured by the number of open grid cells to cover and agent time steps. ANT is an artificial neural network (ANN) that simulates neuromodulation to create a sparse, variable neural network topology and employs an adaptive activation function, with trainable parameters.[1,2]

In the ANT swarm, no central controller exists. The agents do not communicate among themselves, do not have a map, and are not aware of the existence of other agents. An agent appears as an obstacle via the robot/spacecraft sensor. From a central station, the agents receive global information about time, reference directions and total area coverage. The agents do rely on pheromones/markers to track whether a module or location in the target structure has been inspected or covered. In the ANT computer simulations, the agents are constrained by a limited number of time steps, and have no a priori knowledge of the target environment or structure. All the foregoing features simulate a multi-agent swarm with limited, on-board power, computing, memory and sensors.

In the following sections, we describe ANT's novel features, the swarm controller implementation, swarm simulations and test results.

ARTIFCIAL NEURAL TISSUE

In this section, we highlight the novel features of ANT that allow it to outperform classic ANN controllers.[1] Without loss of generality, we assume that agent data/sensor inputs $x_i \in [-1,1]$, neural network connection weights $w_i \in [0,1]$ and activation function outputs $\{-1,1\}$. These values can be other discrete or continuous Real values. We use the terms presynaptic/postsynaptic to indicate which nodes (i.e. artificial neurons) precede/follow, respectively, in a feedforward neural network.

Simulated Neuromodulation

ANNs with simulated neuromodulation have performed well in small-scale, robotic control problems.[1,2,3,4,5] Inspired by neuromodulation in the brain, ANT simulates the release of (chemical) neuromodulators in the neural network layers to determine which nodes comprise the feedforward (motor) neural network that generates the agent output behavior.[2] The ANT neural network layers consist of two types of nodes, the decision and motor nodes, which are located in the same 3D lattice constituting the neural network layers.[2] The neural network activation process consists of two steps. First, all decision nodes receive data/sensor inputs. In our simulations, each spacecraft (i.e. agent) has 26 sensors. If a decision node is excited (e.g. has an activation function output of -1), then the decision node releases a figurative chemical neuromodulator that envelopes the decision node in 3D diffusion zone. If several decision nodes become excited, then their diffusion zones will overlap, and certain lattice cells in the neural network will have high neuromodulator concentrations. If a motor node is in such a lattice cell, then that motor node comprises the feedforward (motor) neural network that generates the agent output behavior.[2] Like a traditional feedforward neural network, all motor nodes in the input layer receive data/sensor inputs, the activation function outputs flow through the feedforward (motor) neural network and the output layer nodes generate the agent output behavior.

Each motor node is endowed with one of eight possible output behaviors listed in Table 1. A simple weighted average scheme of output layer nodes' behaviors determines the agent output behavior.[2] The highest weighted average behavior equal to, or greater than a threshold value of 0.5 is executed. If the agent output behavior is both a turn behavior {1,5,6,7,8} and move behavior {2,3,4}, the turn behavior is executed first. The agent assumes that it is always pointed (agent) reference direction east, and the behaviors in Table 1 are relative to this agent reference direction.

Since different decision nodes can excite during each new set of sensor inputs, different motor nodes can comprise the feedforward (motor) neural network. Hence, the simulated neuromodulation generates a sparse and variable (motor) neural network topology.[2]

Table 1. Motor Node Output Behaviors.

Behavior	Motion (relative to agent reference direction)
1	turn north
2	move northeast
3	move east
4	move southeast
5	turn south
6	turn southeast
7	turn random {S,SE,N or NE}
8	turn northeast

Adaptive Activation Function

The ANT activation function is a linear combination of four step functions shown in Figure 1, and defined more fully in Equation (1) below [2],

$$\Phi(\sigma) = (1-K_1)\left[(1-K_2)\Phi_{down} + K_2\Phi_{up}\right] + K_1\left[(1-K_2)\Phi_{ditch} + K_2\Phi_{mound}\right] \qquad (1)$$

$$\sigma = \frac{\sum_{i=0}^{n} x_i \, w_i}{\sum_{i=0}^{n} |x_i|}$$

where activation function output $\Phi \in \{-1,1\}$, step functions $\Phi_{down}, \Phi_{up}, \Phi_{ditch}, \Phi_{mound}$ are shown in Figure 1, trainable parameters $K_1, K_2 \in \{0,1\}$ and $\theta_1, \theta_2 \in [0,1]$, weighted input σ, n is the number of presynaptic nodes into the postsynaptic node, node inputs $x_i \in [-1,1]$, $x_0 = 1$, weights $w_i \in [0,1]$ and bias weight $w_0 \in [0,1]$.[2] Given the binary parameters K_1, K_2 dominant in Equation (1), the activation function output Φ reduces, and is equal to one of the component step functions in Figure 1.

Swarm Controller Implementation

The ANT swarm controller implemented in this paper substantially follows the robotic controller in Reference 2. The ANT swarm controller consisted of 3 neural network layers, where each layer could contain a possible maximum of 20 × 20 nodes, or a possible total maximum of 1,200 decision nodes and motor nodes in the neural network layers. The initial number of decision and motor nodes is instantiated randomly.

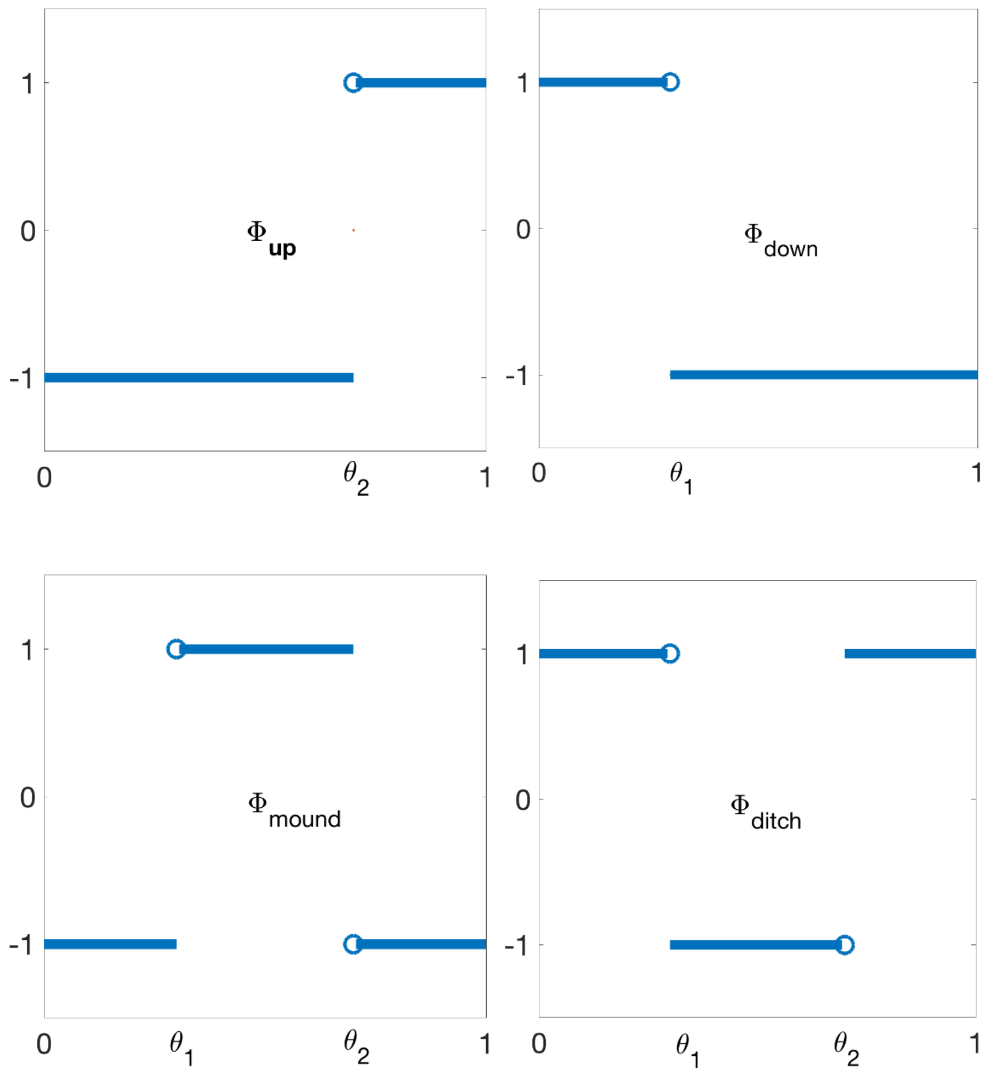

Figure 1. Component Step Functions for ANT Activation Function

All postsynaptic motor nodes can receive a maximum of 9 presynaptic node outputs from a 3 × 3 grid of motor nodes on the prior feedforward (motor) neural network layer, whose center motor node aligns on the same axis as the postsynaptic motor node. All decision nodes and all input layer motor nodes receive data/sensor inputs from the 26 sensor inputs listed in Table 2.

An ANT swarm controller is instantiated randomly and trained/optimized by a simple genetic algorithm (SGA). All trainable parameters for a controller (or an individual) are contained in a symbolic genome.[2] The genome contains one decision gene per decision node, where each decision gene contains 34 trainable parameters total, in which 27 are weight parameters for the sensor inputs. Also, the genome contains one motor gene per motor node, where each motor gene for each input layer motor node contains 34 trainable parameters total (like a decision gene), and each motor gene for the remaining motor nodes contains 17 trainable parameters total, in which 10 are weight parameters for the presynaptic node outputs. A tissue genome exists for a population of controllers (or individuals) and contains parameters that determine how the SGA evolves the genome. A population contains 100 randomly instantiated controllers (or individuals). An evolutionary algorithm (EA) run means a population that has been trained or tested.

The SGA used a Monte Carlo approach, where the swarm attempts to complete an entire coverage/inspection task in one attempt and calculates a fitness score for this attempt. The SGA used a rank-based, roulette selection method to mate (i.e. crossover) individuals among the top half of all fitness scores. The resulting children may undergo possible mutation (one to three percent mutation rate) and possible insertion of a new (daughter) motor genes at a rate of 25 to 75 percent in each generation. If mating/crossover does not occur between two individuals (in the top half of fitness scores due to a crossover criterion[2]), the parent genomes mutate at a rate of 10 to 20 percent to generate children. The population fitness score is the best fitness score among the 100 individuals in the population. To simulate the coverage/inspection of a large space structure, a team of identical (nonholonomic) agents are endowed with the same ANT swarm controller and attempt the following tasks.

SWARM COMPUTER SIMULATION

For a team of homogeneous agents, the simulated task is to cover the 2D open grid areas in Figures 2 – 7, as completely and efficiently as possible. Except for the triangular grid area in Figure 4 with three agents, all other figures use a team of four agents to cover the target area.

In these figures, the blue cells are unvisited open cells, the yellow cells are boundaries, and the cyan cells with colored quivers are cells with agents at their start positions. The direction of the quiver from the cell center to the exterior is the direction of the agent. An agent always assumes to be pointed (agent) reference direction east, and only one agent can be in a cell at any time. When an agent moves from one cell to another, the prior cell becomes cyan to indicate that the cell has been covered/visited by an agent.

Area coverage and the global fitness score are defined as

$$\textit{area coverage} = \frac{U_{beg} - U_{end}}{U_{beg}} \tag{2}$$

$$\textit{global fitness score} = \textit{area coverage} \cdot \textit{visit award} - \frac{R}{U_{beg}} \cdot \textit{revisit penalty}$$

where U_{beg} is the number of beginning unvisited open cells, U_{end} is the number of ending unvisited open cells, R is the total number of cells revisited by the agents during the task, *visit award* = 1 and *revisit penalty* = 0.5. A maximum possible fitness score of 1.0 means that the swarm (or team of agents) covered the target area completely and as efficiently as possible (i.e. with no cells revisited). If the agents did not move in the simulated task, the default fitness score was -0.5.

During each time step, each agent received the 26 sensor inputs listed in Table 2, where on-board sensors provided 23 sensors inputs with binary values {-1,1}, and a central base station provided 3 sensor inputs with continuous values [-1,1].

Table 2. Data/Sensor Inputs to Agent.

Sensor(s)	Sensor data
1-8	Short-range obstacle detector {-1,1}
9-16	Short-range visit detector {-1,1}
17-19	Long-range front obstacle radar {-1,1}
20-22	Long-range front visit radar {-1,1}
23	Agent global reference direction [-1,1]
24	Total global area coverage [-1,1]
25	One-bit memory, prior obstacle encountered {-1,1}
26	Percentage of maximum allowable steps taken [-1,1]

During each training generation, the swarm attempts to cover the target area in one attempt. The number of allowable time steps per agent was limited to force the SGA to find a solution with the lowest complexity with respect to the number of open grid cells to cover and agent time steps. During each time step, the order of which agent moved first was random.

Each EA run was trained for a maximum of 500 SGA generations on Figures 2 – 5 and 1000 generations in Figures 6 and 7. If a population fitness score exceeded for ten generations a fitness threshold of 0.90 to 0.95, training was halted early. The swarm simulation was written and tested in Matlab 2019a on a local Windows 10 (64-bit) workstation with an Intel Xeon CPU, and trained on Intel Xeon university clusters, using Matlab C executables (.mex) generated with GCC 6.3.0 compliers.

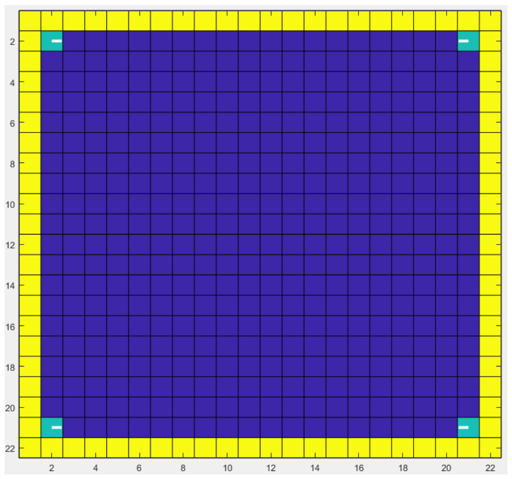
Figure 2. Square Area with Agents in Corners.

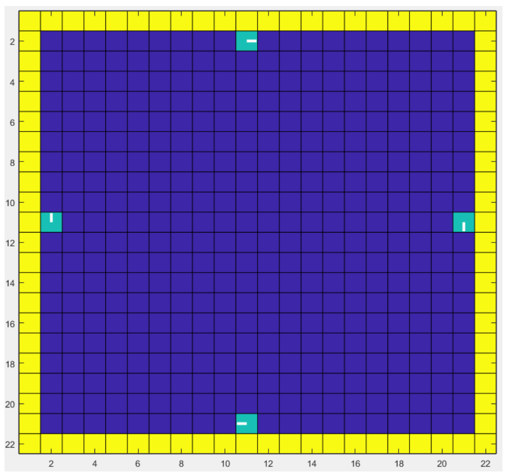
Figure 3. Square Area with Agents at Sides.

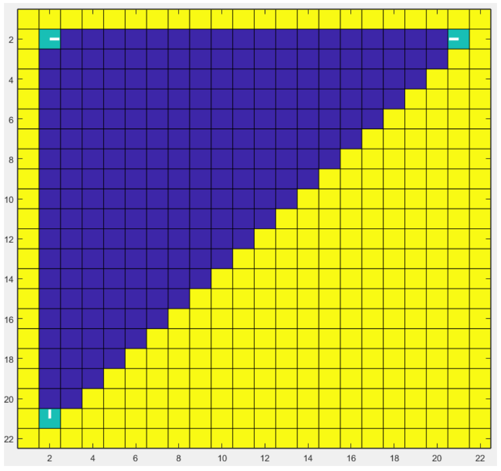
Figure 4. Triangular Area with Agents in Corners.

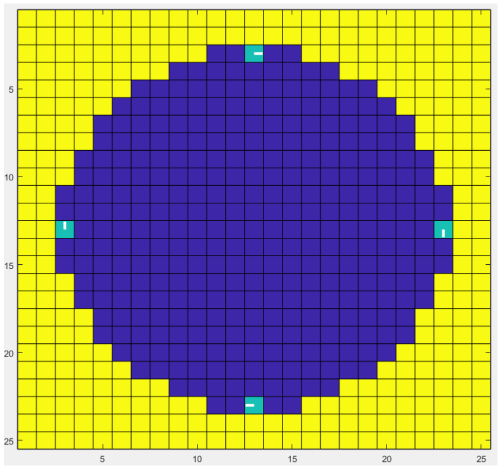
Figure 5. Circular Area with Agents at 3, 6, 9, 12 O'Clock

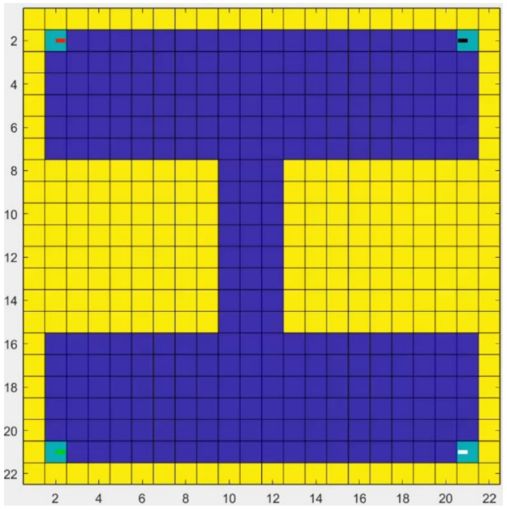
Figure 6. Barbell Area with Agents in Corners

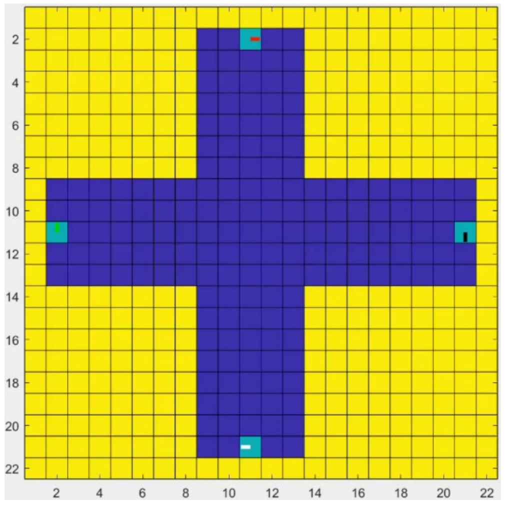
Figure 7. Cross Area with Agents at Ends

SIMULATION TEST RESULTS

Using the final evolved populations trained by the SGA, Table 3 presents the baseline test results for 100 EA test runs on each figure. The Table 3 columns titled Agents, Open Cells, Total Steps mean, respectively, the number of agents/spacecraft in the swarm, the beginning number of unvisited open grid cells to cover (exclusive of the cells occupied by agents at their starting positions) and the number of agents times the number of time steps allowed for each agent in the simulation. The Area Coverage and Global Fitness Score values are defined by Equation (2).

On each figure (or grid area), the total number of time steps by the swarms were limited in testing (as in the training episodes) to model limited on-board power for the agents. Except for the circular area (Figure 5) and the barbell area (Figure 6), the ANT swarms covered the grid areas completely. For the circular area (Figure 5) with 329 unvisited open cells at the beginning, only 13 unvisited open cells remained at the end of the simulation. For the barbell area (Figure 6) with 264 unvisited open cells at the beginning, only 2 unvisited open cells remained at the end of the simulation.

Except for the circular (Figure 5), barbell (Figure 6) and cross (Figure 7) grid areas with quasi-linear times, the ANT swarms achieved essentially, linear time complexity, with respect to the number of open cells to cover and agent time steps. The ANT swarms on the barbell and circular areas produced the lowest fitness scores of 0.95 because the agents needed to revisit some open cells or turned in-place to cover these grid areas, similarly on the cross grid area. On all these figures, the swarms coordinated to move in the same clockwise/counterclockwise direction along the area exteriors and spiraled inwards. For example, see Figures 8 and 9 for the barbell and cross areas, respectively, at time steps number 17 and 14 (for each swarm agent), respectively. For the barbell area, notice in Figure 6 that the agents begin pointed toward each other and in Figure 8, the agents are moving clockwise about the area exterior.

Table 3. ANT Swarm Area Coverage Results Over 100 EA Runs

Grid Area	Agents	Open Cells	Total Steps	Area Coverage				
				Mean	Medium	Std.	Max.	Min.
Square Figure 2	4	396	400	0.96	0.98	0.04	1.0	0.70
Square Figure 3	4	396	400	0.92	0.92	0.05	1.0	0.75
Triangular Figure 4	3	207	240	0.78	0.90	0.23	1.0	0.27
Circular Figure 5	4	329	400	0.88	0.89	0.06	0.97	0.67
Barbell Figure 6	4	264	280	0.86	0.91	0.15	0.99	0.28
Cross Figure 7	4	175	200	0.77	0.74	0.10	1.0	0.51

Table 4. ANT Swarm Fitness Scores Over 100 EA Runs

Grid Area	Agents	Open Cells	Total Steps	Global Fitness Score				
				Mean	Medium	Std.	Max.	Min.
Square Figure 2	4	396	400	0.95	0.97	0.05	1.00	0.65
Square Figure 3	4	396	400	0.88	0.88	0.07	0.99	0.64
Triangular Figure 4	3	207	240	0.74	0.84	0.24	0.99	0.26
Circular Figure 5	4	329	400	0.82	0.86	0.09	0.95	0.49
Barbell Figure 6	4	264	280	0.85	0.91	0.15	0.95	0.26
Cross Figure 7	4	175	200	0.69	0.64	0.13	0.98	0.47

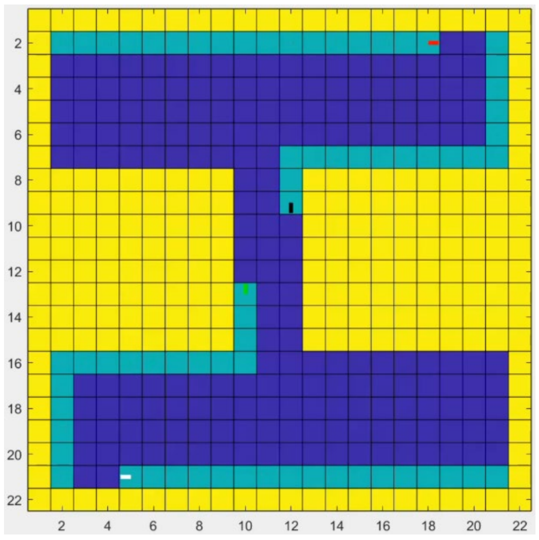

Figure 8. Agents at Time Step 17 in Barbell Area

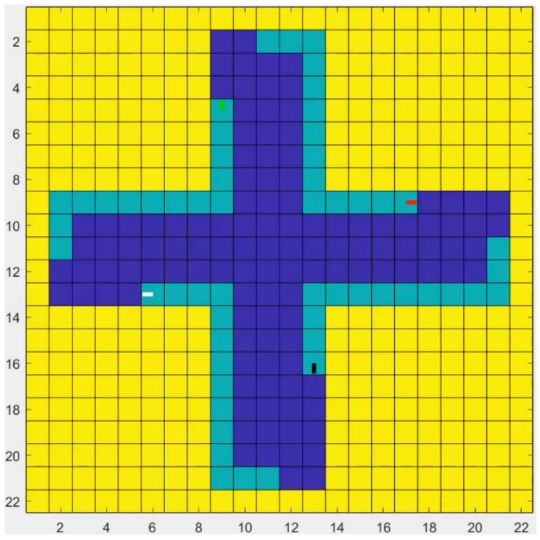

Figure 9. Agents at Time Step 14 in Cross Area

CONCLUSION

In summary, we presented computer simulations of the ANT controller that can coordinate a swarm of identical agents to cover 2D basic geometric shapes completely, or near completely, in linear or quasi-linear time complexity, with respect to the number of open grid cells to cover and the agent time steps. Although many swarm algorithms exist, most of these algorithms are not practical for real-world use because they need substantial human intervention, rely on hardwired path planning, and cannot operate under the real-world constraints of limited on-board power, computing, memory and sensors. Also, many algorithms rely on a central controller.

On the other hand, the ANT swarm controller is decentralized and can operate with limited on-board power, computing, memory and sensors, and achieve near optimal area coverage in computer simulations. Hence, ANT is an attractive controller to consider for spacecraft swarms to automate the inspection of large structures assembled in space. Given the ANT swarm results, we hope in the future to conduct high fidelity dynamics simulations, followed by demonstrations on robotic hardware in laboratory.

REFERENCES

[1] J. Thangavelautham and G. M. T. D'Eleuterio, "Tackling Learning Intractability Through Topological Organization and Regulation of Cortical Networks," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 23, no. 4, pp. 552–564, Apr. 2012.

[2] J. Thangavelautham, K. Law, T. Fu, N. A. E. Samid, A. D. S. Smith, and G. M. T. D'Eleuterio, "Autonomous multirobot excavation for lunar applications," *Robotica*, vol. 35, no. 12, pp. 2330–2362, Dec. 2017.

[3] P. Husbands, "Evolving robot behaviours with diffusing gas networks," in *Evolutionary Robotics*, Berlin, Heidelberg, 1998, pp. 71–86.

[4] A. Philippides, P. Husbands, T. Smith, and M. O'Shea, "Flexible Couplings: Diffusing Neuromodulators and Adaptive Robotics," *Artificial Life*, vol. 11, no. 1–2, pp. 139–160, Jan. 2005.

[5] S. Cussat-Blanc and K. Harrington, "Genetically-regulated Neuromodulation Facilitates Multi-Task Reinforcement Learning," in *Proceedings of the 2015 on Genetic and Evolutionary Computation Conference - GECCO '15*, Madrid, Spain, 2015, pp. 551–558.