# Improving the Efficiency and Scalability of Multi-Drone Coverage Systems with Decentralized Control

Adit Shah,[1†] Bryce L. Ferguson,[2] Jason R. Marden[2]
[1]The Athenian School, 2100 Mt. Diablo Scenic Boulevard, Danville, CA 94506
[2]Department of Electrical and Computer Engineering, University of California, Santa Barbara, CA 93106
†Email address: aditshah00@gmail.com

UC SANTA BARBARA Research Mentorship Program

THE ATHENIAN SCHOOL

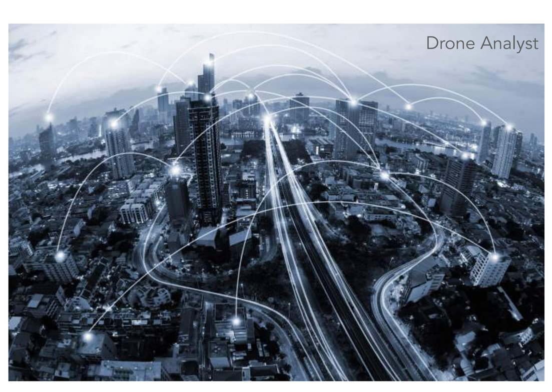
## Abstract

Multi-agent systems are effective for numerous applications because they complete tasks more resiliently and efficiently than monolithic systems. This paper focuses on a sensor-coverage problem in which a limited fleet of sensor-equipped drones must survey an area and extract maximum information. Applications include environmental monitoring, disaster relief, and surveillance systems. Traditionally, these systems are implemented through a centralized approach, which can run into obstacles, including communication and computational constraints. These limitations can be mitigated through decentralized control algorithms, where the decision-making and navigation processes are localized to individual robots. In this paper, we look to quantitatively compare algorithms for such decentralized systems through simulation. We establish a baseline with a greedy algorithm, and we then compare the performance to that of a log-linear learning approach, which adds stochasticity. Finally, we propose a method to automatically generate a sensitivity constant for this algorithm, and verify its performance.

## Multi-Drone Coverage Systems

### The Problem

- Survey a large area with sensor-equipped drones to extract as much information as possible, even if insufficient drones to cover entire area [1]
- Minimize coverage overlap
- If drones can not cover entire region, prioritize areas with higher utility (importance)





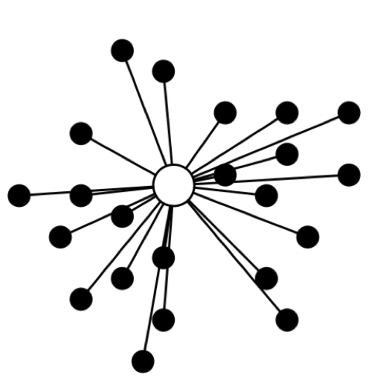
### Applications

- Surveillance and security systems
- Environmental monitoring, e.g. endangered species counts
- Surveying the extent of a disaster (oil spill, wildfire, etc.)
- Mapping a city or neighborhood, inaccessible rural areas
- Inspecting dangerous compounds

### Control Approaches

#### Centralized Approach

- One node controls all drones, including position and navigation
- Easy to control, regulate the system because only one decision-maker [2]
- Coverage problem is NP-complete: centralized approach is inefficient [3]



#### Decentralized Approach

- Each drone tries to optimize its own location, avoids other drones obstacles on its own
- More efficient, scalable than centralized approach
- Cannot always find global optima

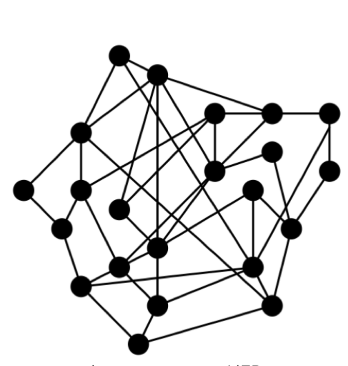


Image courtesy: AIER

## Research Objectives

1. Develop multi-drone simulation
2. Implement and compare algorithms
3. Develop and verify enhancements

## System Model

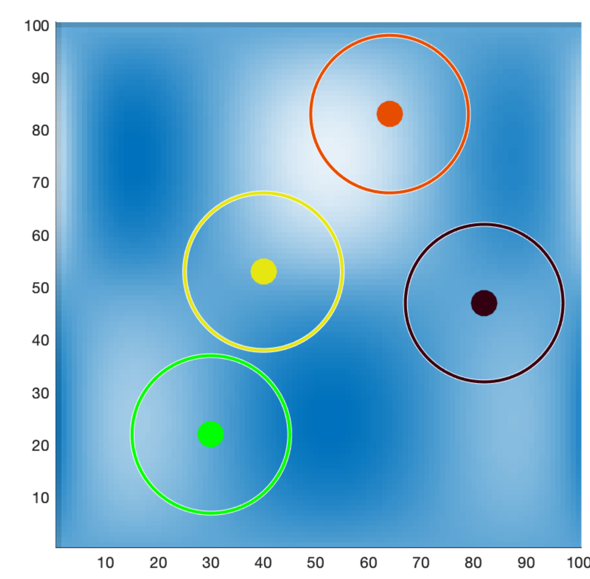



Fig. 1: Map of sample surveillance area $M^1$

- Fig. 1 shows an example map, $M^1$, of a surveillance region of size $p$ by $q$
- The value of every cell $M_{x,y}$ is the utility, or importance of covering cell $(x, y) \in X \times Y$ where $X = \{1,\ldots,p\}$ and $Y = \{1,\ldots,q\}$
- Darker cell = higher utility
- Each agent (represented by colored dots) starts at a random or specified position $a_i(0)$
- The colored circle around each agent illustrates its sensing radius $R$
- The coverage set of each agent is defined as:

$$(x, y) \in X \times Y \mid d(a_i, (x, y)) \leq R$$

- At each time-step $t$, an agent $i$ is chosen at random to make a move $a_i \in A_i(t)$ from within its movement radius $r$:

$$A_i(t) = (x, y) \in X \times Y \mid$$
$$d((x, y), a_i(t-1)) \leq r, (x, y) \notin a_{-i}(t-1)$$

- The utility $U_i$ of any possible action $a_i \in A_i(t)$:

$$U_i(a_i, a_{-i}(t-1)) = \sum\nolimits_{(x,y)\in C(a_i)\setminus\bigcup_{j\neq i} C(a_j)} M_{x,y}$$

- The goal of the decentralized algorithms is to maximize the total system's utility $\mathscr{U}_t(a)$:

$$\mathscr{U}_t(a) = \sum\nolimits_{(x,y)\in C(a(t))} M_{x,y}$$

## The Greedy Algorithm

- Greedy algorithms have been widely studied in the context of such set-cover problems for multi-agent systems
- In the greedy algorithm, each agent solely aims to maximize its own utility
- An agent will only make a move if the new location has a higher utility value
- An agent's next move $a_i(t+1)$ is chosen as follows:

$$a_i(t+1) = \arg\max_{a_i \in A_i(t)} U_i(a_i)$$

- Terminates when $a_i(t) = a_i(t+1)$ for a number of iterations twice the total number of agents
- Will get stuck at local optima, even only takes actions with immediate benefit

## Log-Linear Learning

- Choses an action $a_i(t+1) \in A_i(t)$ through a probability distribution: probability of making action $a_i \in A_i(t)$ with temperature $\tau > 0$ is defined as follows [4]:

$$p_i^{a_i}(t) = \frac{e^{\frac{1}{\tau}U_i(a_i, a_{-i}(t-1))}}{\sum_{\overline{a}_i \in A_i} e^{\frac{1}{\tau}U_i(\overline{a}_i, a_{-i}(t-1))}}$$

- $\tau$ determines likelihood that $i$ will make a suboptimal action
  - As $t \to 0$, acts as a greedy algorithm
  - As $t \to \infty$, selects actions at random
- Intuitively, $\tau$ is an"exploration" constant, where higher values encourage agents to search for global optima beyond local optima near its initial position
- $\tau$ decays by 0.997 at each iteration

## Algorithm Comparison

Table 1:
Random starting positions

| | | $t$ | $\mathscr{U}_t(a)$ | $\tilde{\mathscr{U}}_t(a)$ |
|---|---|---|---|---|
| Greedy | $\tau = 0$ | 54.7 | 2082.3 | 0.4257 |
| LLL | $\tau = 40$ | 570.0 | 2110.3 | 0.4315 |
| | $\tau = 96$ (auto) | 675.3 | 2177.0 | 0.4451 |
| | $\tau = 130$ | 700.0 | 2131.7 | 0.4358 |

Table 2:
Starting position in one corner

| | | $t$ | $\mathscr{U}_t(a)$ | $\tilde{\mathscr{U}}_t(a)$ |
|---|---|---|---|---|
| Greedy | $\tau = 0$ | 110.7 | 1907.7 | 0.3900 |
| LLL | $\tau = 153$ | 771.0 | 2174.7 | 0.4446 |
| | $\tau = 192$ (auto) | 753.3 | 2215.0 | 0.4529 |
| | $\tau = 250$ | 1125.3 | 2141.0 | 0.4377 |

Tables 1 and 2. Performance metrics for the greedy and log-linear learning algorithms (with varying $\tau$ values) on $M^1$. These simulations were run with $n = 6$ agents, sensing radius $R = \sqrt{5}$ and moving radius $r = 1$. Three trials per setting were conducted and the average values of $t$ (time-steps to stabilization), $\mathscr{U}_t(a)$ (system utility), and $\tilde{\mathscr{U}}_t(a)$ (normalized system utility) are displayed.

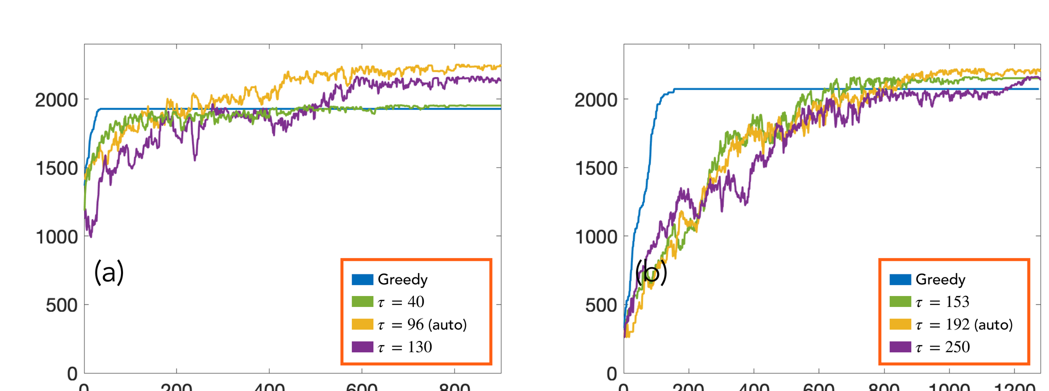



Fig. 2. Sample graphs of $\mathscr{U}_t(a)$ (vertical axis) over $t$ (horizontal axis) for $M^1$ for each algorithm. (a) shows a random starting position, (b) shows a corner starting position.

- Greedy algorithm: achieves lowest $\mathscr{U}_t(a)$
  - Worse performance for corner starting position, because higher chance of encountering local optima
  - Converges in fewest time-steps
- LLL with Automatic Tau Generation: highest $\mathscr{U}_t(a)$
- $\tau$ > generated value: converges faster, lower $\mathscr{U}_t(a)$
  - Occurs because agents do not cross all local optima
- $\tau$ < generated value: converges slower, lower $\mathscr{U}_t(a)$
  - Unexpected result – more exploration time should lead to higher utility
  - Possible explanation: high $\tau$ values act essentially at random

## Automatic Tau Generation

- Log-linear algorithm in [1] does not specify how to select $\tau$
- Manual tuning is necessary – can be automated
- Factors that correlate most with ideal $\tau$ are sensing radius, average utility, and starting configuration
- Formula for random starting position ($\overline{M}$ is average value of $M$)

$$\tau = \frac{1}{2}\pi R^2 \cdot \overline{M}$$

- Formula if agents begin in one corner (needs to be scaled up because of higher chance of encountering local optima)

$$\tau = \pi R^2 \cdot \overline{M}$$

## Conclusion & Future Work

### Summary

- Formulate a modified version of a decentralized multi-agent coverage problem
- Simulate this multi-robot system and compare various algorithms in this simulation: greedy and log-linear learning
- Implement automatic $\tau$ generation algorithm
- Demonstrate that log-linear learning with automatic $\tau$ generation has the best performance

### Future Work

- Test algorithms on maps with different more/fewer local optima, other characteristics
- Additional starting configurations (e.g. all agents begin at the center, or in "groups" at all four corners)
- Analyze effect of different variables
  - Sensing radius, movement radius, number of agents, distance from other agents, and adding obstacles
  - Take into account for $\tau$ generation
- Verify algorithms on multi-robot systems

**Acknowledgements**

The first author would like to thank lab peers Siddharth Ganesan and Nischal Sinha for their support throughout the program. We would like to acknowledge Center for Computation and Dynamical Control for access to their resources and support during this research. Finally, we thank Dr. Lina Kim, Michael Hughes, and A S M Iftekhar from the Research Mentorship Program, and the University of California, Santa Barbara, for the research opportunity.

**References:**

[1] G. Chmaj and H. Selvaraj, "Distributed Processing Applications for UAV/drones: A Survey," Progress in Systems Engineering. Advances in Intelligent Systems and Computing, vol. 366, pp. 449–454, 2015.

[2] E. Galceran and M. Carreras, "A survey on coverage path planning for robotics," Robotics and Autonomous Systems, vol. 61, no. 12, pp. 1258–1276, Dec. 2013.

[3] K. Loayza, P. Lucas, and E. Pelaez, "A centralized control of movements using a collision avoidance algorithm for a swarm of autonomous agents," 2017 IEEE Second Ecuador Technical Chapters Meeting (ETCM), 2017.

[4] J. R. Marden and J. S. Shamma, "Revisiting log-linear learning: Asynchrony, completeness and payoff-based implementation," *Games and Economic Behavior*, vol. 75, no. 2, pp. 788–808, Jul. 2012.