APIS II: Extension and Flight Testing of the Autonomous Persistent Intelligent Swarm Algorithm

Principal Investigator: John E. Pye (Honors Capstone)

Faculty Advisors: Prof. George Halow, Prof. Vasileios Tzoumas

Research Team: John E. Pye (Honors Capstone), Ashley Ruder, Blake Marlow, Sarah Nseir, Sara Gregg, Lania Brown, Joseph Steiman, Nadia Poopat, Brenden Kashat, Gavriil Stavrakas, Ian Christensen, Matthew Kroll, Abi Parivakkam, Anthony Galan, Bryce Halterman, Matthew Kroll

Abstract

Multi-agent Cyber-Physical-Human (CPH) systems in realistic environments operate under uncertain conditions. Task inaccuracies, both in terms of exact location and exact priority, are to be expected. Furthermore, real-world deployment can encounter communication range limitations, leading to disconnected swarm components over short and medium periods. The APIS algorithm, first presented in [1], addresses this issue through a bee-foraging heuristic approach to task completion, balancing direct task completion (harvesting), mapping (dancing), and informational interchange (dancing). We propose to implement the Autonomous Persistent Intelligent Swarm (APIS) system in a real-world heterogeneous swarm setting, evaluating its efficacy in maintaining lines of communication, reducing performance uncertainty, and maintaining swarm cohesion.

We will undertake two parallel efforts: First, improving the fidelity of the APIS simulation and evaluating its mission efficacy in various system configurations. Second, creating a physical APIS-compatible system capable of completing various tasks and mapping the environment to validate the simulation results and serve as a proof-of-concept technology demonstrator.

Introduction and Overview

The opportunities for effective heterogeneous multi-agent systems have been the subject of extensive recent research. Collaborative swarms can perform highly effective mapping and localization [1][2], energy-efficient environmental inspection [3], and

agricultural activities [4], to name a few applications. Towards this end, research has focused on swarm consensus-building for efficient tasking [5], safe task allocation [6], trajectory planning [7], and other extensions of traditional autonomy and control to multi-agent systems. However, this work assumes excellent connectivity, known environments, or centralized coordination. This paradigm is reasonable in small spaces or tight formations [8] but cannot extend to more expansive spaces where connectivity is no longer guaranteed.

APIS, proposed in [9], presents a method to reduce uncertainty in heterogeneous multiagent systems by adopting Honeybee Foraging (HBF) behavioral characteristics. Results demonstrated increased effectiveness and decreased variability compared to similar (but non-heuristic) methods. However, these results were obtained through low-fidelity simulation, so their applicability in real-world situations is untested and unverified. Therefore, we propose implementing the proposed APIS behaviors in a real system, deployed first in a laboratory test environment and ultimately in a controlled, real-world environment, to evaluate its fitness and make such upgrades necessary to facilitate actual operation.

Our primary objective is to better understand the performance tradeoffs involved in altering heuristic task weightings, so that in the future we might increase APIS reliability by implementing optimality guarantees applied to submodular problems as described in [10] and [11]. Improving our simulation fidelity will also allow for us to implement an automatic swarm weighting tuner, aiming to optimize the swarm's weighting parameters to adapt to the operational environment.

A secondary objective of our work is to bridge the sim-real gap and test APIS in increasingly high-fidelity environments. The key difference between the assumptions made in simulation and real-world deployment is communications. As our model communication protocol, we will deploy the BATMAN (Better Approach to Mobile Ad-hoc Networking) mesh networking protocol specified in [12]. This way, we will capture the typical limitations and considerations of mesh networking when testing APIS. In theory, such mesh networking protocols will allow for the same assumed capabilities as those relied upon in [9]. However, the now-increased size of the operational area compared to the simulation area and the potential for radio-blocking structures not accounted for in the simulation introduces the need for dedicated communication node agents. The introduction of network repeater agents will also relax the communication equipment requirements of the swarm, which will enable us to

incorporate more commercial off-the-shelf (COTS) components. These strategically positioned agents will allow for lines of communication to be formed around abstractions and across large distances.

Although APIS, as presented in [9], showed initial promise, it lacks some of the optimization parameters introduced in [13] and [14], especially concerning Age of Information (AOI) and bandwidth. These parameters help to describe communication backbone quality, which is assumed to be constantly good in [9], so long as agents are within range of one another. Furthermore, the applications of the submodular problem definition outlined in [15 – 19] present an ideal model on which to base our problem definitions. Although the problem suggested in [9], wherein agents patrol an environment inspecting known objects of interest, is essentially submodular, it is not a wholly accurate example problem. Multiple agents are barred from contributing to ongoing tasks, even in a limited capacity, whereas most real-world tasks permit multiple-agent assignments. Task positions and initial priorities are known to the swarm, whereas in the real world, some tasks are likely in unknown positions or only defined by a general region. All agents can accomplish all tasks, thus limiting the application of heterogeneity. Therefore, a tertiary objective of our current research is to create a new set of example tasks bounded by submodularity but more like real-world task analogs to allow for a more holistic demonstration of the capabilities and limitations of APIS.

Goal

This capstone aims to take the APIS algorithm, developed by the author while at NASA, and prepare it for flight testing. Thus, we strive to:

1. Adapt APIS and comparison algorithms for real-world flight testing in the context of several tasks accomplished by UAVs.
2. Upgrade our APIS simulation framework to better capture the real behavior of the physical swarm in our flight testing.
3. Analyze simulation results to study tradeoffs inherent in tuning swarm heuristic weightings.
4. Implement APIS on flight hardware and demonstrate some base level of operability to identify key challenges in real-world deployment.

Preliminary Definitions

Let us define n sets of drone classes, $C_{n\in Z}$ with each class C_n containing i drones, so that we can define each drone as $C^{i\in Z_n}_{n\in Z_s}$, denoted as the ith drone of the nth class, where Z_n is the integer set of the drones comprising a class of drones, and Z_s is the integer set of classes consisting of the swarm of drones.

Let us assume each drone $C^{i\forall Z_n}_{n}$ is homogeneous but distinguishable, with identical initial parameters and capabilities. Let us also assume that each class $C_{n\forall Z_s}$ is heterogeneous compared to the other class and may have similar or different initial parameters and capabilities. This creates a heterogeneous drone swarm with homogeneous constituent drone classes. This is a reasonable assumption, as it mirrors many real-world systems, like commercial aircraft fleets.

Let us define some initial parameters and capabilities for each class. Drone classes can be distinguished based on their flight time, communications range, mission capabilities, movement capabilities, current and future state.

Let us define t^i_n as the current flight time remaining of the ith drone in the nth class, which is identical among all drones $i\in Z_n$ at the start of operations, so we can say that $t^{i\forall Z_n}_{n\exists Z_s}(0) = k_n$ where k_n is a constant representing the initial flight time of the nth class where $n\in Z_s$.

Let us define r_n as the communications range of all drones in the nth class, which is identical among all drones $i\in Z_n$ during operations.

Let us define Ξ_n it as the set of mission capabilities of all drones in the nth class, which is identical among all drones $i\in Z_n$ during operations. Let us also define M it as the set of all missions present in the environment, m_j representing the $j$mission in the environment $m_j\in M$. Thus, a drone i within a class n can only complete a task m_j if $m_j \in \Xi_n$, which is to say, the task m_j is within the set Ξ_n of all tasks completable by a

drone in class n. We will dub this relationship "task ability", whereby $m_j \in \Xi_n$ we can say that a task j is taskable by class n.

Let us define U_n as the set of physical capabilities of all drones in the nth class, which is identical among all drones $i \in Z_n$ during operations. The drone i exerts a control effort $u_i \in U_n$ to bring it to a new state to move. Let us also introduce $\hat{U}_n^i$ as the set of *safe* control efforts for the drone i in class n. While we will define what we mean by "safety" later, let us temporarily assume that it is a strict requirement $u_i \in \hat{U}_n^i$ and, thus, is a safe control effort. We will dub this u_i as $\hat{u}_i$ the safe control effort derived from the desired control effort.

Let us define $X \in R^3$ as the set of all attainable three-dimensional states within the environment. Let us, as before, also define $\hat{X} \in R^3$ as the set of all safe states within the environment - for example, states that are not too close to any obstacles or not too high or too low in altitude. By definition of the variables, if an agent i starts in a safe state $\hat{x}_i(0)$ and only exerts control efforts $\hat{u}_i$, then at all times $t_{current}$ during the mission, $\hat{x}_i(t_{current}) \in \hat{X}$, meaning that the system maintains a safe position concerning all obstacles.

Finally, let us define $F_{n \exists Z_s}$ representing the dynamical model of the nth class of drones, where $F_{n \exists Z_s}$ is identical for all drones i in class n. We can find the state evolution of agents in the swarm by evaluating $F_n(u_{i \exists Z_n}(t)) + x_{i \exists Z_n}(t) \rightarrow x_{i \exists Z_n}(t + \Delta t)$ where $u_{i \exists Z_n}(t)$ is the current control effort of the agent i of class n, F_n is the model associated with that class, $x_{i \exists Z_n}(t)$ is the current state of that agent, and $x_{i \exists Z_n}(t + \Delta t)$ is the new state of that system.

Let us assume that the models $F_{n \forall Z_s}$ are known to all agents in the swarm. This is generally a strong assumption due to the inherent nonlinearities of robotic systems and environmental disturbances. Therefore, let us relax this assumption by assuming that

$F_{n\forall Z_s}$ is merely sufficient to predict the movement of agents in the interstices between received message packets, carrying the states of agents as reported by those agents. This relaxed assumption may also be overly strong; we will examine exactly how reasonable it is in this paper by analyzing (1) how often agents receive first-person (reported by the agent in question) state updates, (2) how long it takes for those messages to be received, and (3) the level of accuracy of $F_{n\forall Z_s}$ which is necessary to enable proper operation of the APIS algorithm, through the correct prediction of agent states and tasking preferences.

Swarm Architecture

Let us consider five cases of the swarm in operation. In each case, we will deploy a swarm of three classes and a ground station, constituting a fourth class nominally but otherwise incapable of tasking.

Drone Classes

Class 1: Heavy Drone



Figure 1: Prototypical Heavy-Lift Drone in use in the M-LAPIS (Michigan-Leidos Autonomous Persistent Intelligent Swarm) Flight Testing Project

Heavy drones represent a class of drones defined by their large and expensive payload. In our case study, the heavy drones will be equipped with Velodyne VLP-16 Puck Lite, which can survey surrounding environments and provide high-accuracy localization measurements. Heavy drones represent the least numerous swarm classes due to their cost. They are not, however, capable of completing other primary mission tasks due to the diversion of all available payload capacity to the LiDAR.

Class 2: Medium Drone



Figure 2: Prototypical Medium-Lift Drone, in use in the M-LAPIS (Michigan-Leidos Autonomous Persistent Intelligent Swarm) Flight Testing Project

Medium drones represent a class of drones defined by their unique, mission-tailored payloads. In our case study, the medium drones will be equipped with visual and thermal cameras and software-defined radio receivers, able to search the surrounding environment for objects of interest (AprilTag fiducial markers, thermal anomalies, and radio sources). Due to the allocation of payload capacity towards mission-capable sensors, medium drones can complete main mission tasks. Therefore, medium drone numbers are primarily dictated by the number of tasks in the environment.

Class 3: Light Drone



Figure 3: Prototypical Light-Lift Drone in use in the M-LAPIS (Michigan-Leidos Autonomous Persistent Intelligent Swarm) Flight Testing Project

Light drones represent a class of drones defined by their small size, limited capabilities, and role in extending the mesh network. In our case study, light drones will be equipped with minimal sensors but the same communications network hardware as all other drones, enabling the extension of communications without deploying more expensive assets. Due to their limited size and lack of effective mission payload, light drones cannot complete main mission tasks.

Class 4: Ground Station



Figure 4: Example Ground Station Laptop

Despite the autonomy of the drone swarm, we want to account for the reality that in real-world operations, a level of human oversight capacity is required. Therefore, we introduce a non-agent swarm member, the ground station, which only monitors the

swarm by collecting the duplicate packets as other agents receive. Thus, connectivity with this agent will factor into the overall scoring of connectivity of the swarm and will be optimized to improve connectivity but will not contribute to completing any tasks.

Swarm Implementation and Tasks

In order to validate the capability of APIS to coordinate drones in the real world, we created three different classes of drones to demonstrate heterogeneous task assignment and coordination. We deploy one Heavy Drone, two Medium Drones, and three Light Drones for our implementation. Each drone is equipped with a mesh network WiFi transceiver, and each class has a defined set of tasks it can complete. We use a radio beacon direction-finding task, a visual search task, and a thermal source search task to create realistic tasks with different required sensor payloads. For our initial implementation, however, we will focus on the radio, visual tasks, and Heavy and Medium drones. All drone classes are currently operational, and all task completion methods have been simulated. Our next step is to move from simulation to reality and deploy the drones in an all-up flight test.

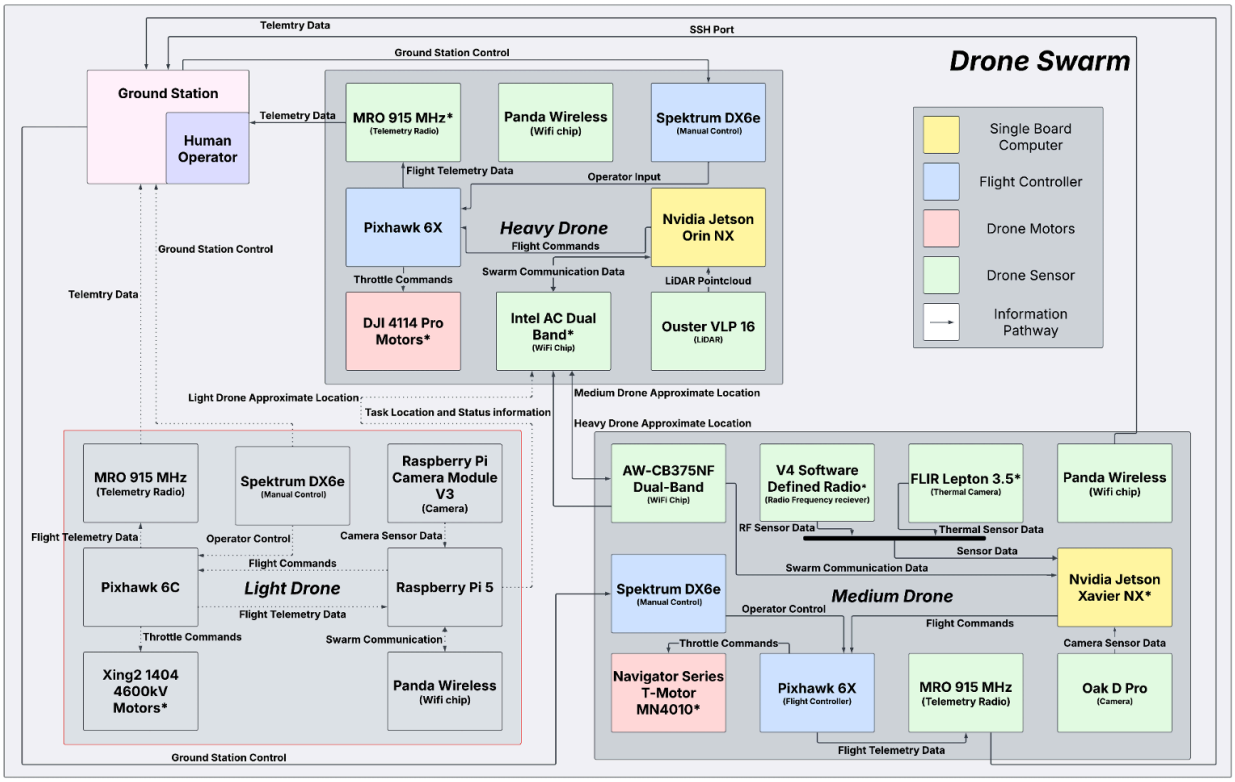


Figure 5: Drone swarm physical and logical architecture, with currently de-scoped light drone.

We have designed our testing swarm to be modular. Therefore, we can add and remove classes of drones at will, with minimal reconfiguration of autonomous behaviors. We demonstrated this functionality by removing our Light Drone class during preliminary trials, with this reduced architecture depicted in Figure 5. As we move towards more complex flight testing, we will be able to re-integrate this class to extend the operational range of our swarm.

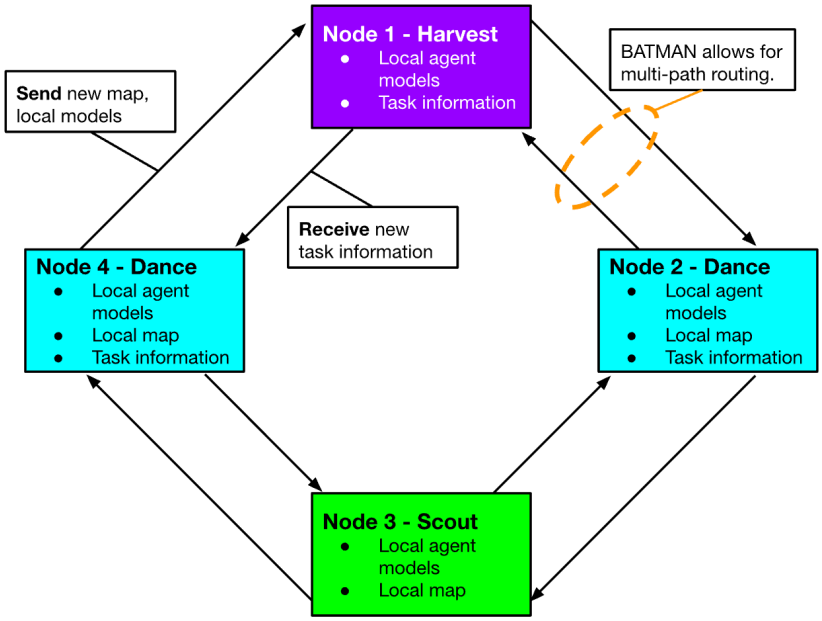


Figure 6: Mesh network swarm architecture, with three jobs (harvest, scout, dance).

The swarm's reconfigurability and modularity is based on the flexible communication links provided by the BATMAN mesh networking protocol. In Figure 6, we can see that a heterogeneous system of agents, each with different data generation and intake requirements, is able to communicate over the mesh network. Furthermore, the automatic re-routing capabilities of the protocol enable the swarm to change roles as dictated by the onboard APIS algorithms, without placing undue burden on the communications infrastructure. Based on this reconfigurable architecture, five cases of system operational states emerge.

Case 1: Fully Connected

We first consider the ideal case, where all agents are within communication range of one another.

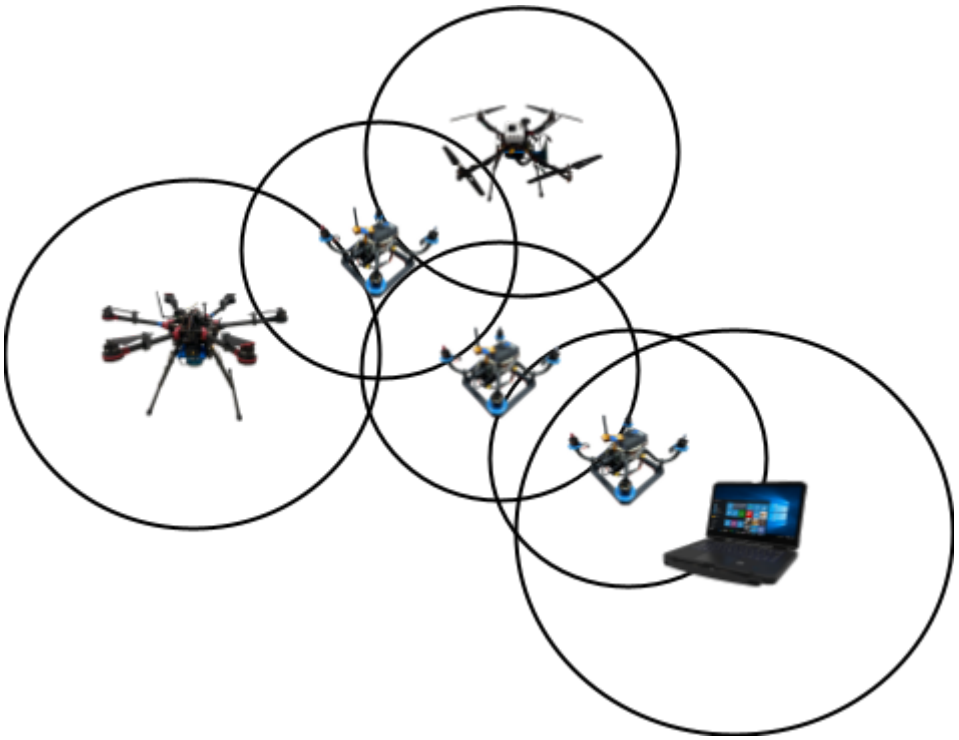


Figure 7: Visual representation of a fully connected swarm.

To aid in swarm analysis, we can represent this as a bidirectional, partially connected graph. In Figure 6, below, we see that one distinct graph has formed, with all nodes connected.

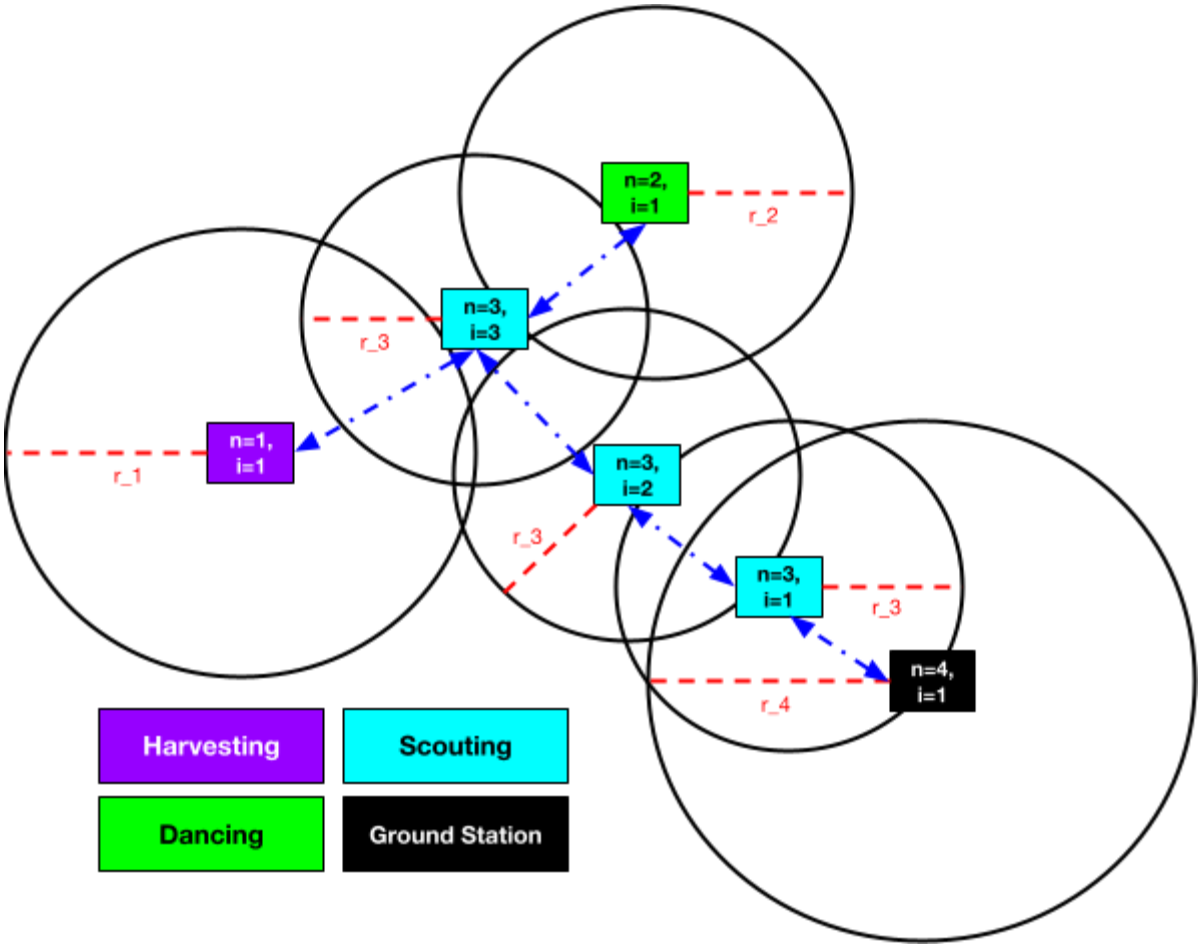


Figure 8: Graphical representation of a fully connected swarm.

We assume Case 1 to be the optimal and steady state of the system. The swarm should converge to a fully connected state when APIS functions appropriately. However, we allow for oscillation between Case 1, fully connected, and Case 3, partially connected. This allows for short-term tasking directives, which may temporarily place agents outside the general communications network. We shall perform our later communications lag analysis on this state, as in all other semi-connected states, the communications lag may be infinite for some perpetually unconnected agents.

Case 2: Unconnected

We next consider the worst case for swarm performance, in which all agents are out of communication range of one another.

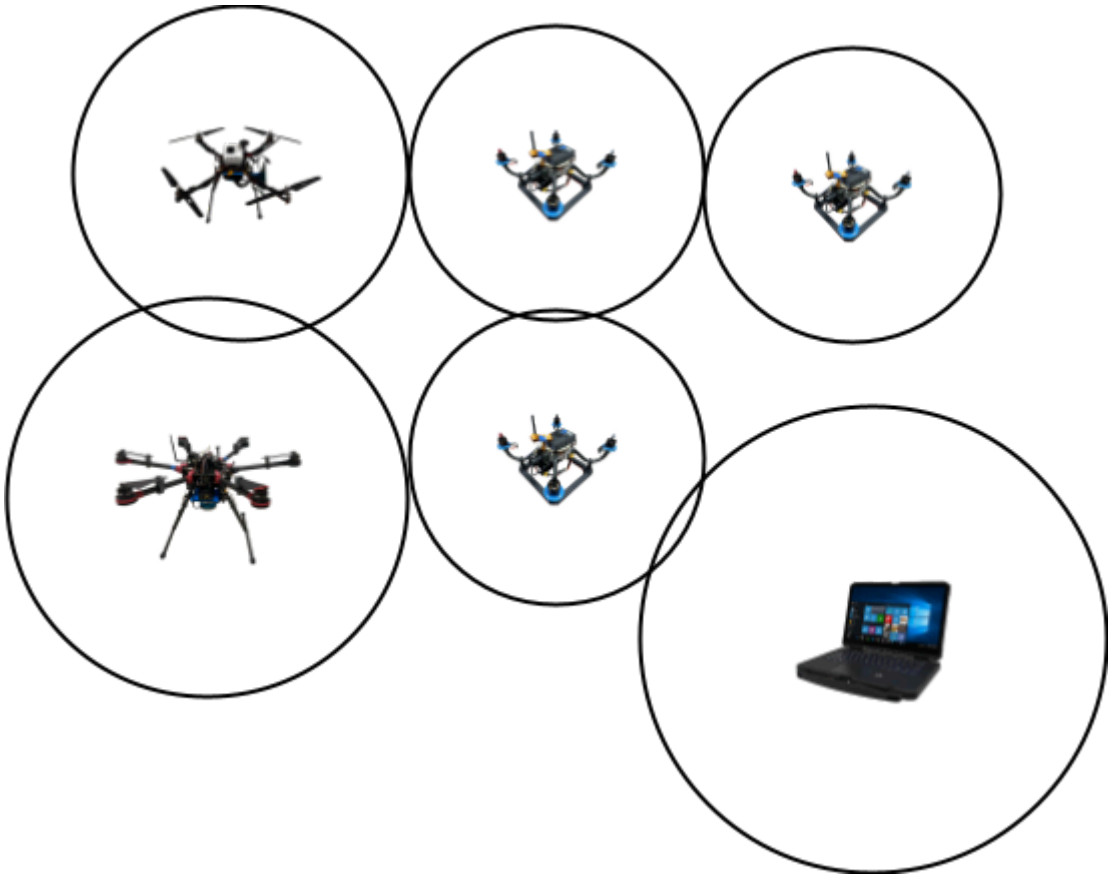


Figure 9: Visual representation of a completely unconnected swarm.

To aid in swarm analysis, we can represent this as a bidirectional, partially connected graph. In Figure 10, we can see that no connected graph has been created, and instead, there are six unconnected nodes.

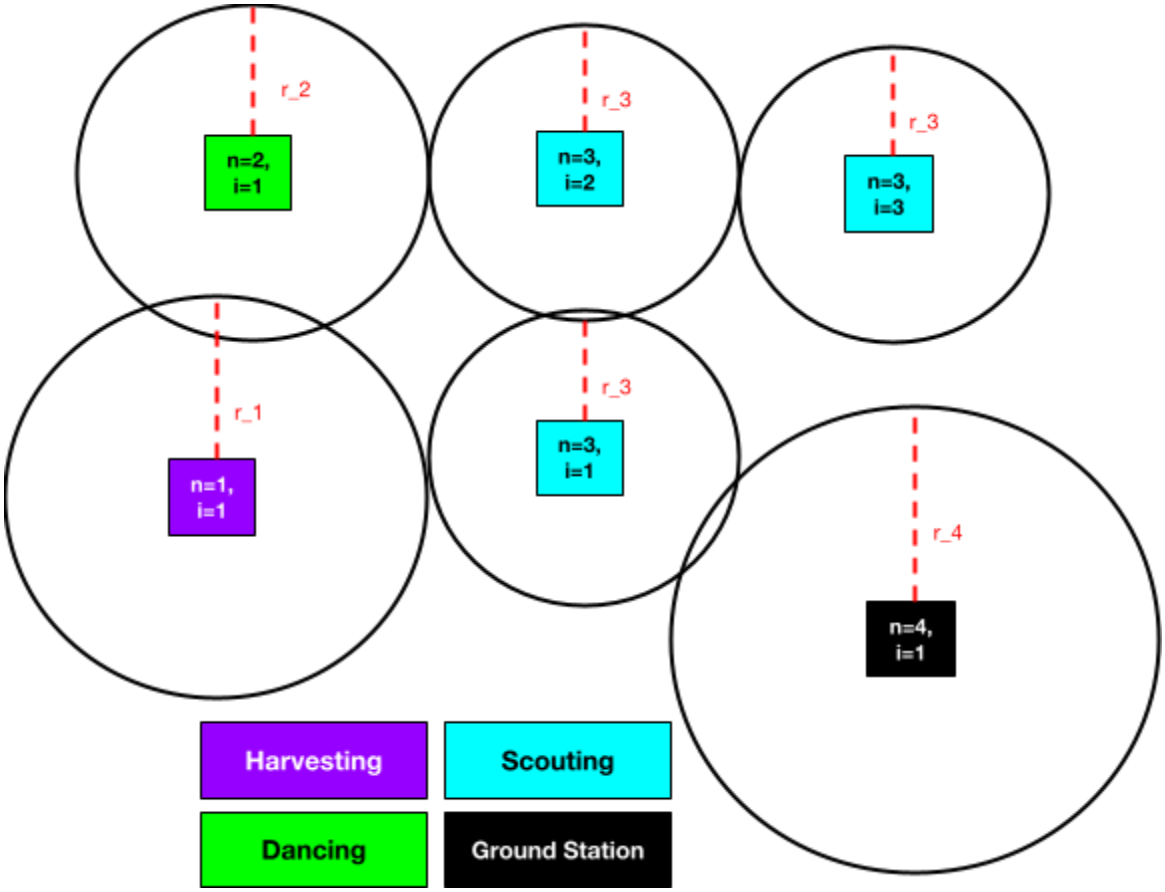


Figure 10: Graphical representation of a completely unconnected swarm.

Case 2 is troubling, as it imposes extended reliance on agents upon their internal memory models; as discussed earlier, we do not assume such models to be perfect, therefore, extended Case 2 operations will slowly reduce the optimality of agent tasking decisions, as they mispredict the behaviors of fellow agents. Under nominal operations, the overriding goal of a system in Case 2 should be to ultimately converge to Case 1 through purposeful movements to bring agents back into communication contact.

Case 3: Partially Connected, in Favor of Connection

Next, we consider a more realistic scenario where most swarm agents are connected.

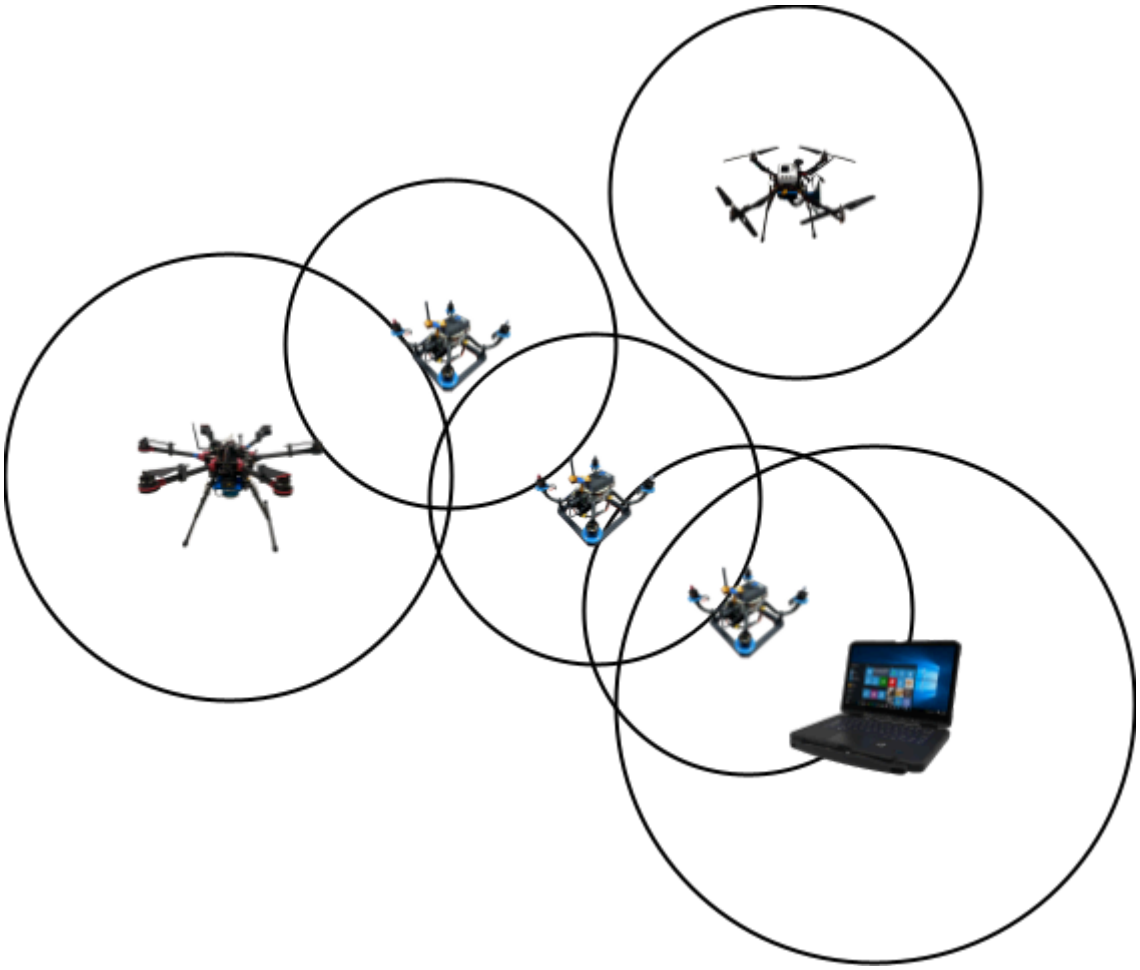


Figure 11: Visual representation of a partially connected swarm with more connected agents than unconnected agents.

To aid in swarm analysis, we can represent this as a bidirectional, partially connected graph. In Figure 10, below, we can see that one distinct graph has formed, along with a single unconnected node on the graph's periphery.

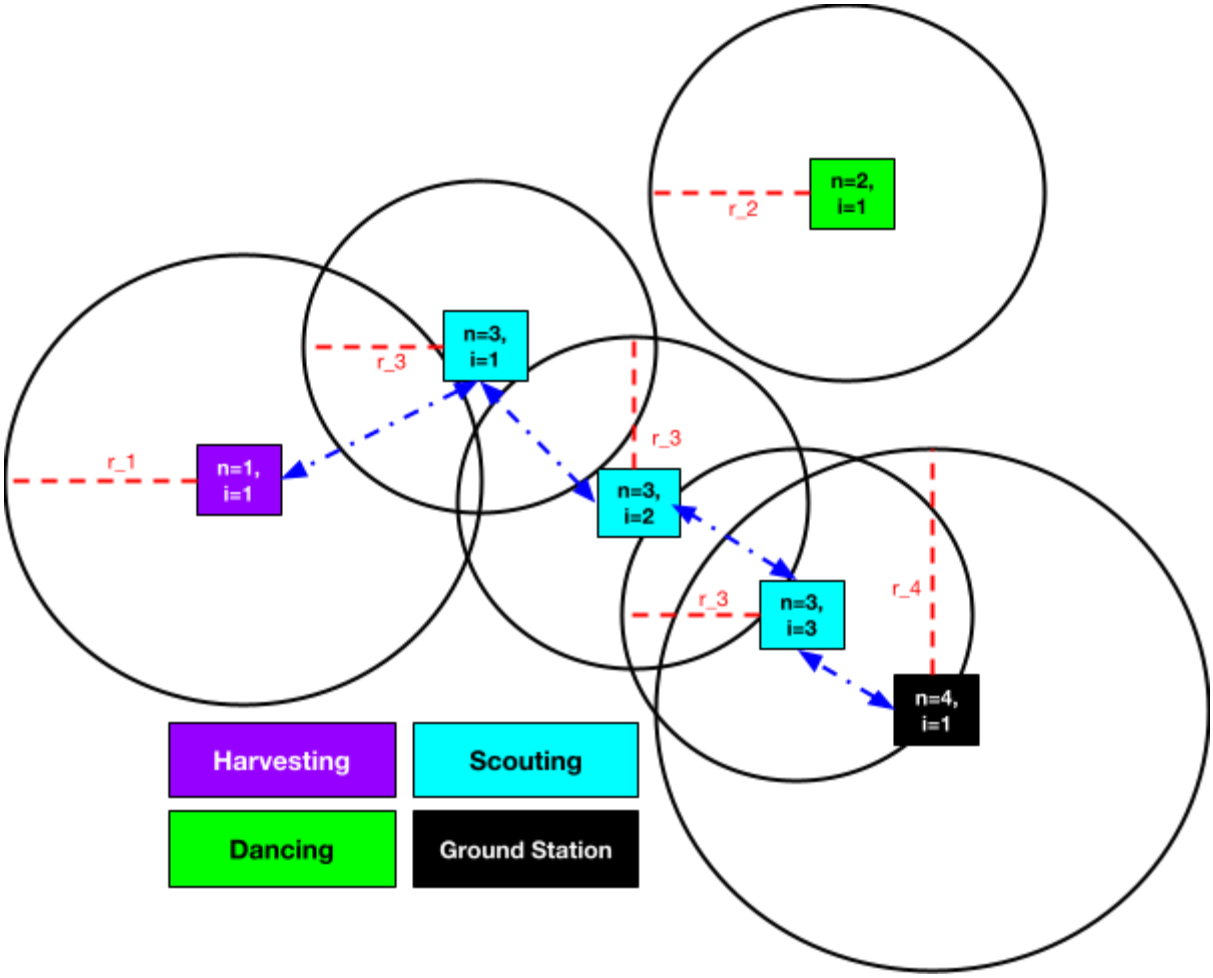


Figure 12: Graphical representation of a partially connected swarm with more connected agents than unconnected agents.

We can also assume Case 3 to be transiently optimal, both as a transition state to Case 1 and as a desirable state on its own merits, depending on the spatial distribution of tasks throughout the environment. We assume Case 3 not to be a steady state, as disconnected agents must ultimately reestablish contact with the wider swarm, both to share the results of their actions and to update internal memory models.

Case 4: Partially Connected, in Favor of Disconnection

We likewise consider a partially connected swarm, where some agents are connected, but most are unconnected.

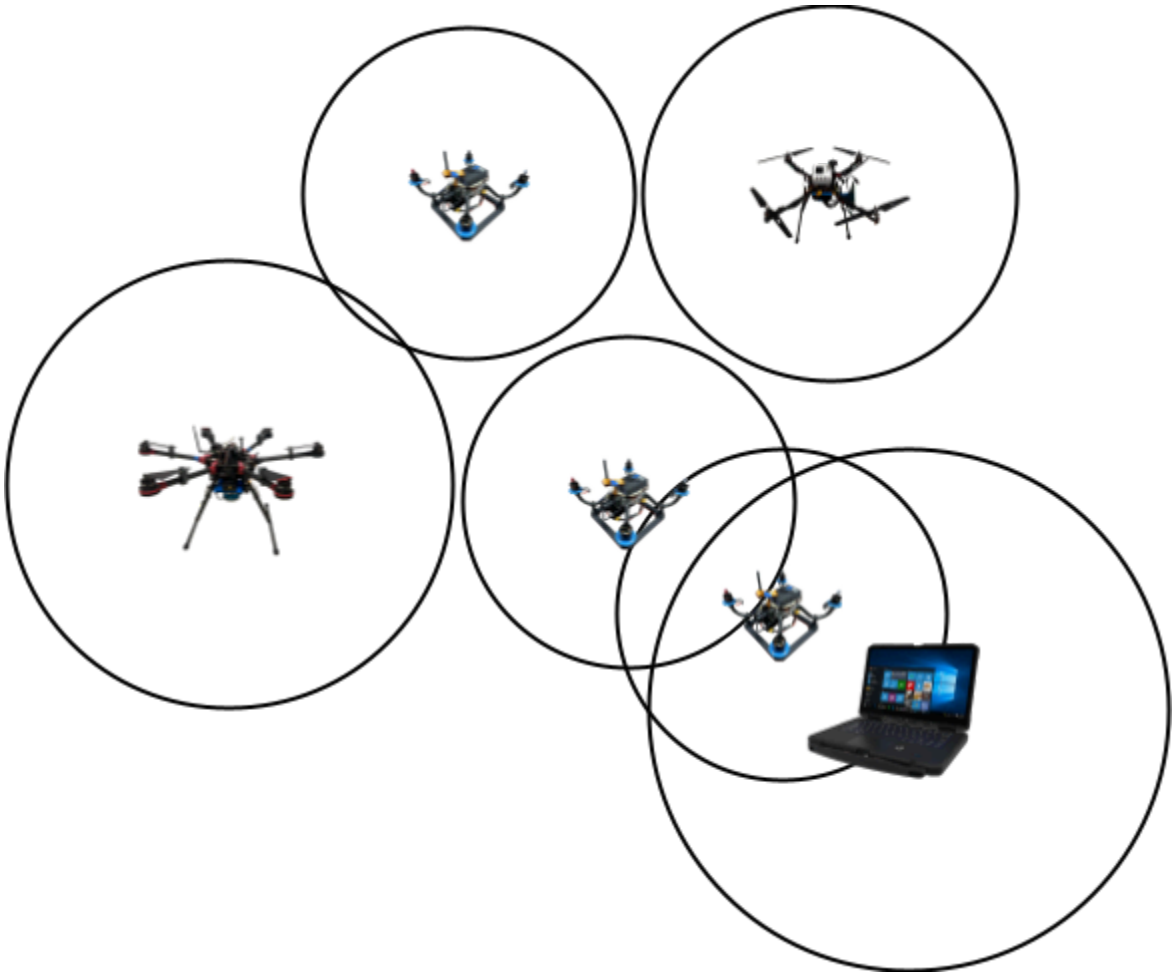


Figure 13: Visual representation of a partially connected swarm with more unconnected agents than connected agents.

To aid in swarm analysis, we can represent this as a bidirectional, partially connected graph. In Figure 12, below, one distinct graph has formed, along with three unconnected nodes on the graph's periphery.

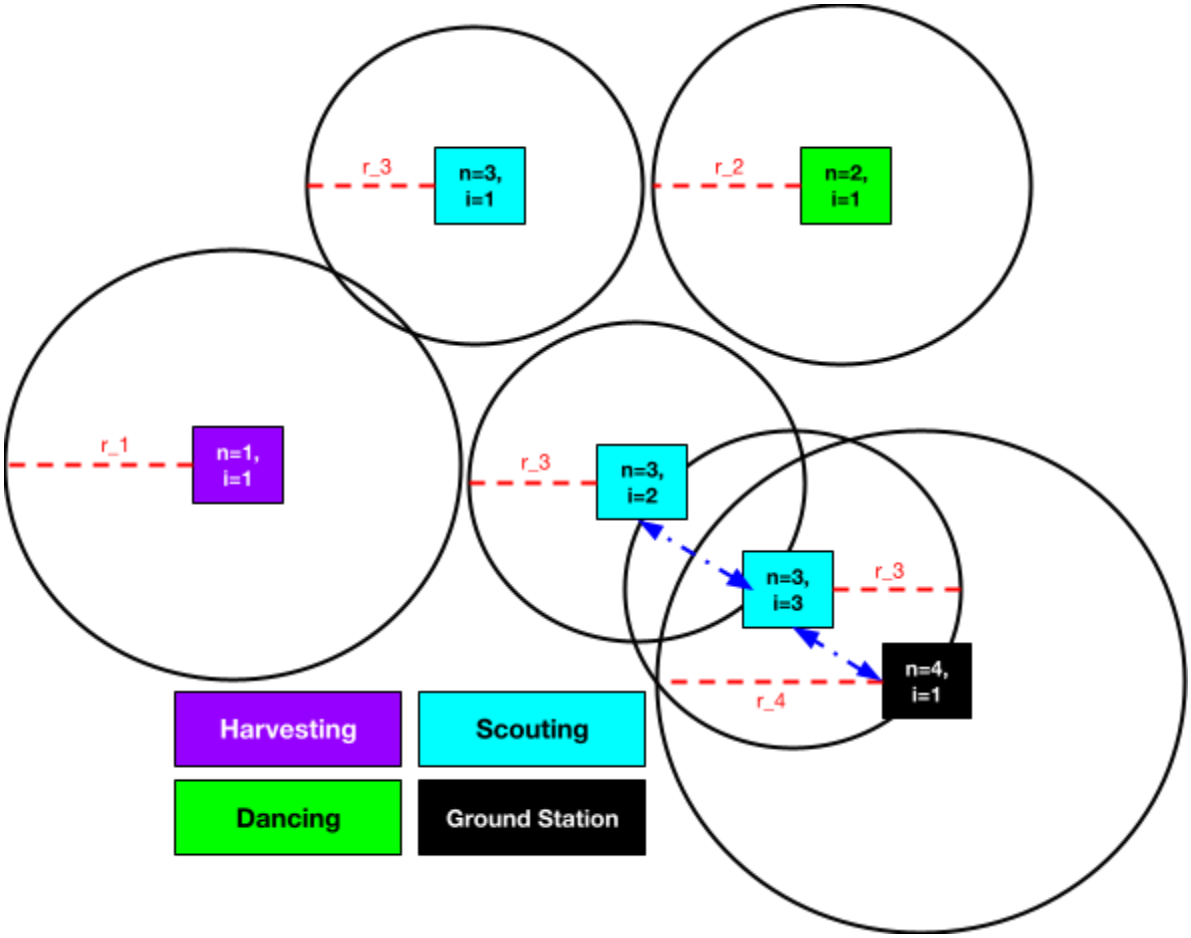


Figure 14: Graphical representation of a partially connected swarm with more unconnected agents than connected agents.

Case 4 is almost certainly a transition state based on the wider system operations concept. By its nature, it is unstable, as all unconnected agents following APIS protocols ought to prioritize re-establishing contact. Case 4 is typical in real-world operations, as it normally represents the system's initial state.

Case 5: Drone Cliques

We finally consider an interesting edge case in which agents are connected into several distinct “cliques,” or unconnected groups of communicating agents.

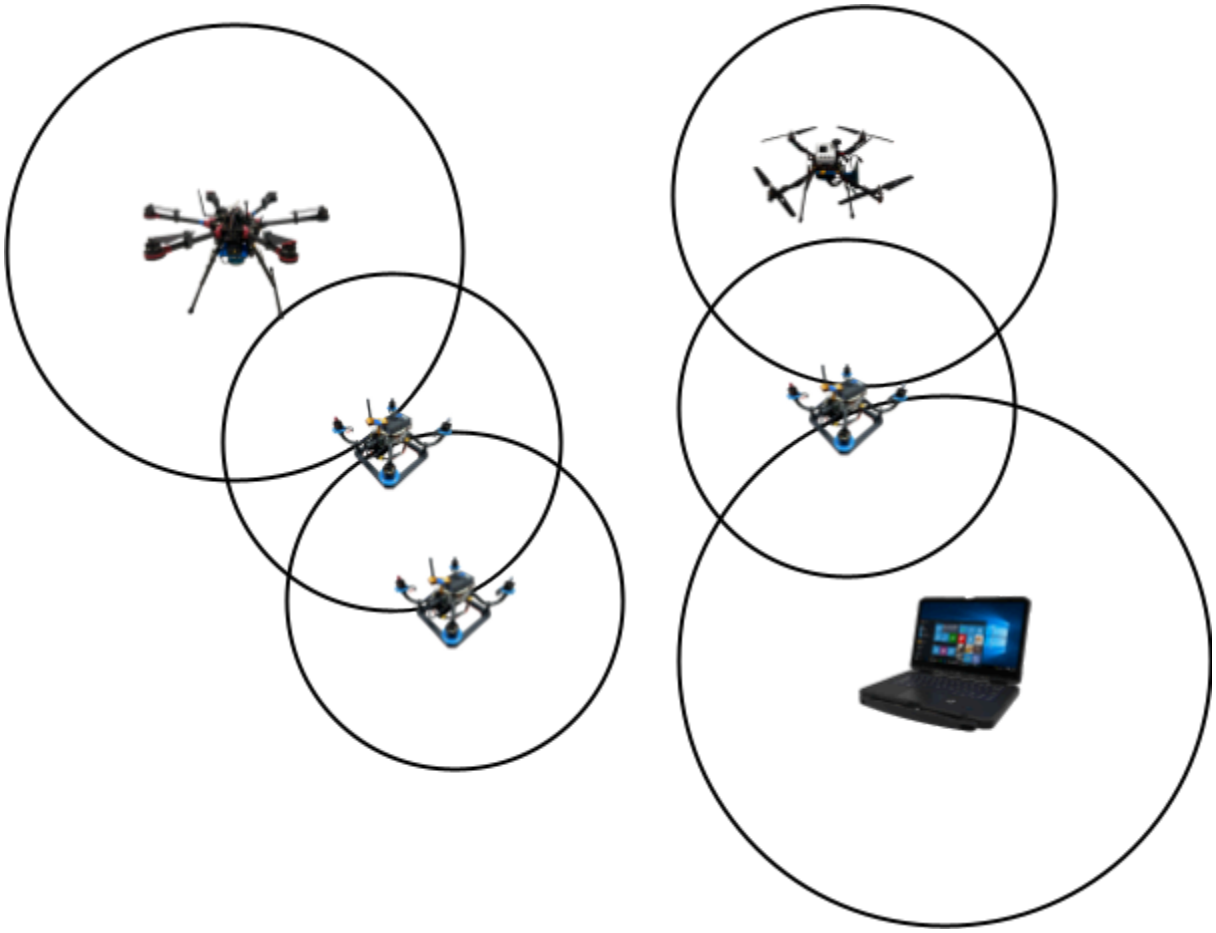


Figure 15: Visual representation of a two-clique swarm, where all agents have gathered into two groups that do not communicate group-to-group.

To aid in swarm analysis, we can represent this as a bidirectional, partially connected graph. In Figure 16, we can see that two distinct graphs have formed, each representing a clique.

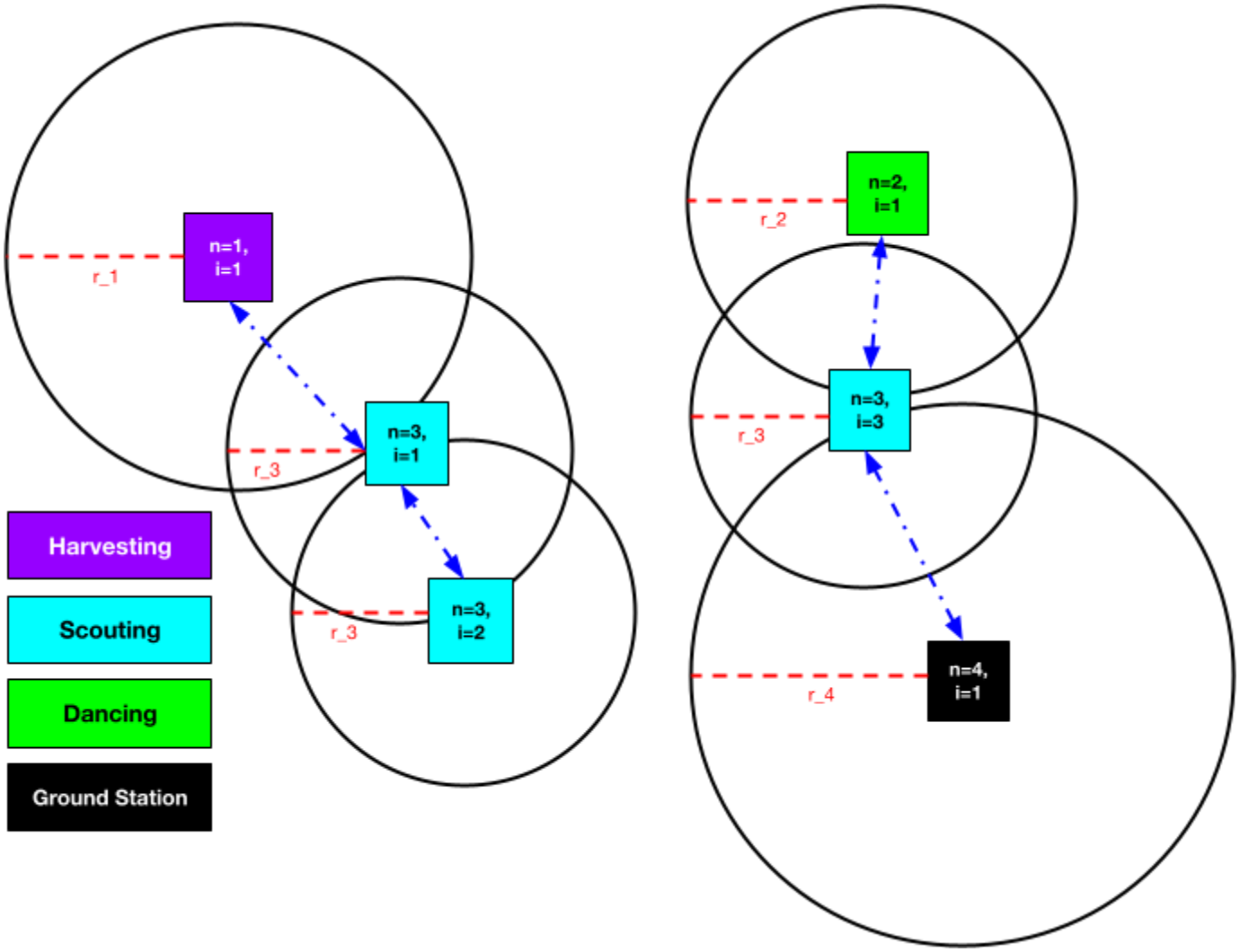


Figure 16: Graphical representation of a two-clique swarm, where all agents have gathered into two groups that do not communicate group-to-group.

Case 5 is the most dangerous state of the system. It is stable, especially in cases where equally-sized cliques form, as seen above in Figure 16. APIS addresses this case by introducing stochasticity into tasking and commands. Thus, it breaks the clique deadlock by forcing a transition into either Case 3 or Case 4 and, ultimately, to Case 1. One of the essential goals of APIS is to minimize the probability of or time spent in Case 5 states.

Task Design

To evaluate APIS's performance in the real world, we set out to create four distinctive tasks that demonstrate the swarm's heterogeneous coordination capabilities.

Table 1: Definitions of tasks for testing in simulation and the real world.

Task	**Explanation**	**Servicing Agents**	**Status**
Fly a survey pattern and collect LiDAR data to make a map.	LiDAR units are heavy and expensive, so only a subset of agents will be outfitted with them. Therefore, it is necessary to task these units to create an environmental map for obstacle avoidance and distribute it to other agents.	Class 1 (Heavy)	Completed and implemented.
Fly a survey pattern and monitor the visual camera data for AprilTag fiducials [19].	AprilTag fiducials provide an ideally recognizable visual pattern, but debris can still obstruct them, requiring a thorough environmental search to locate them.	All classes.	Completed and implemented.
Compare radio signal strength with other agents and fly to the extrapolated position.	Radio beacons can be almost indistinguishable from the environment and require collaboration between agent receivers to locate them. This presents an ideal opportunity to demonstrate submodular tasking amongst agents.	Class 2 (Med.)	Temporarily descoped to only requiring a single drone.
Calculate and go to the area with the highest mesh network potential (lowest network coverage).	Network quality parameters can only be optimized if a dedicated task is assigned. This task provides lightweight but networking-equipped agents with a useful task to increase swarm cohesion.	Class 1 (Heavy), Class 2 (Med.), Class 3 (Light)	Completed and implemented.

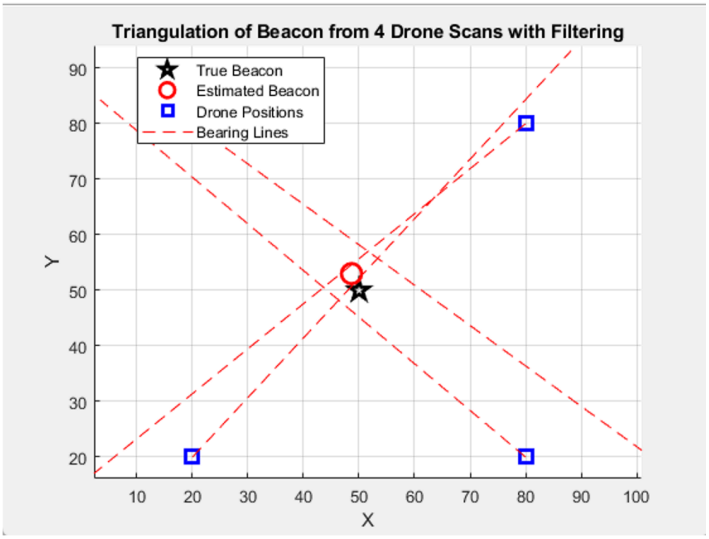


Figure 17: Simulated radio beacon direction-finding, using a directional antenna and a single drone.

Our system's ability to communicate peer-to-peer enables the efficient completion of tasks requiring or benefiting from multiple agents. In the current swarm configuration, the radio beacon direction-finding task only takes advantage of a single agent. However, we plan to extend this task to allow multiple-agent assignment, increasing completion speed by surveying multiple triangulation nodes simultaneously. This task takes advantage of the directional antenna and software-defined radio mounted on one of the medium drones.

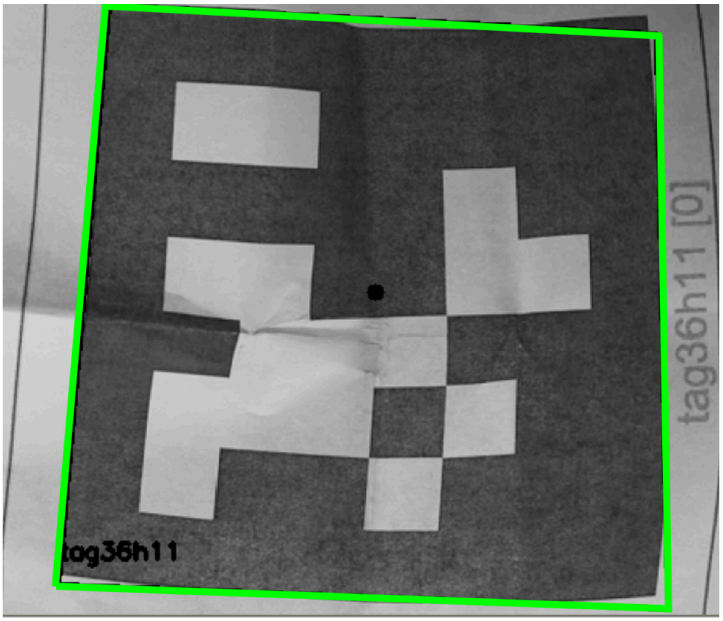


Figure 18: Visual fiducial detection, using an AprilTag.

We also designed a second type of task: for agents to fly a simple area survey pattern to visually detect an AprilTag fiducial. Utilizing a simple onboard visual camera, this task is intended to be completed by any class of drone. So, it represents a tasking conflict with the radio task and provides a “harvesting” alternative assignment for all classes of drones.

APIS Fundamentals

Given the drones $C^{i\in Z_n}_{n\in Z_s}$ and drone characteristics $F_{n\exists Z_s}$, and finally, the tasks in the environment M, we can formulate $P^{i\forall Z_n}_{m\in M}$ the initial priority of the task m for the agent i. Each agent will choose a task to service, changing its priority. It will then select a new task to service. If the priority of the task changes before the agent can complete the task, or other task priorities increase, the agent may re-task to a more opportune target.

The goal of the swarm is to minimize the priority of all environmental tasks. This is equivalent to completing tasks as they arise quickly. A secondary goal of the swarm is to mitigate the trial-to-trial variation of average priority. This represents more predictable tasking, allowing APIS swarms to be deployed into environments with more consistent results.

To evaluate APIS's efficacy, we will compare its performance and variance to other state-of-the-art methods and some archetypical methods. We will also conduct fully optimal tasking, removing all communication and uncertainty restrictions for each random permutation. Therefore, we can evaluate the fractional optimality of our implementation versus other implementations. Finally, we will record the time it takes for the swarm to finish all tasks Ξ, thus yielding a speed metric describing how quickly the swarm was able to complete its mission.

Using our simulations, we will record the sum of all true priorities of all tasks in the environment at each timestep. This priority is based on (1) the urgency of the task (how important it is) and (2) the age of the task. We will average this sum by the number of simulation timesteps, yielding an average proper priority. We will compare this value to the theoretical optimal priority, yielding a fractional optimality metric. We will also compare the average priorities for each trial in a configuration, yielding a variance of the average priority metric. The time completion metric will be recorded as the timestep at which the swarm completed the final task. These metrics will form the basis of comparison between algorithms.

We will begin by restating the fundamental algorithms in [8] but in terms of our problem definition, and with some alterations to reflect our new problem definition.

Algorithm 1: Auctioning Task Assignment with No Commitment, No Coordination

Given:

$C_{n\in Z_s}^{i\in Z_n}$: The set of all drones in the mission area

$t_{n\exists Z_s}^{i\forall Z_n}(0)$: The initial flight times of all drones

r_n: The communications range of all drones

Ξ_n: The set of mission capabilities of the drones in the nth class

U_n: The set of physical capabilities of the drones in the nth class

$F_{n\exists Z_s}$: The dynamical model of the drone in the nth class

M: The set of all current and future tasks in the mission area

Do until $M = \emptyset \vee t_{n\exists Z_s}^{i\forall Z_n} = 0$:

For each class n in C:

For each agent i in C_n:

UnCoordinatedTasking(i)

EvolveTargets()

EvolveAgents()

End Do

Algorithm 1 details the most straightforward approach to task assignment, in which every agent greedily selects the best choice presented. This approach presents several vulnerabilities, where agents multi-assign to tasks, or repeatedly switch tasks without making progress. The EvolveTargets() and EvolveAgents() functions are mission-dependent, simply representing the evolution of system models by a single timestep. UnCoordinatedTasking() represents a greedy tasking approach, wherein each agent will choose the most fit task, as dictated by the shortest-distance task.

Algorithm 2: Auctioning Task Assignment with Full Commitment, No Coordination

Given:

$C_{n \in Z_s}^{i \in Z_n}$: The set of all drones in the mission area

$t_{n \exists Z_s}^{i \forall Z_n}(0)$: The initial flight times of all drones

r_n: The communications range of all drones

Ξ_n: The set of mission capabilities of the drones in the nth class

U_n: The set of physical capabilities of the drones in the nth class

$F_{n \exists Z_s}$: The dynamical model of the drone in the nth class

M: The set of all current and future tasks in the mission area

Do until $M = \emptyset \vee t_{n \exists Z_s}^{i \forall Z_n} = 0$:

    For each class n in C:

        For each agent i in C_n:

            If $i.task$ is None:

                UnCoordinatedTasking(i)

    EvolveTargets()

    EvolveAgents()

End Do

Algorithm 2, although improving in terms of preventing agents from repeatedly and suboptimally switching, still does not address the issue of agents inter-coordinating their tasks to prevent double-tasking. Furthermore, the blanket prevention of premature task switching introduces a new risk of tasks that become suboptimal while still being serviced.

Algorithm 3: Auctioning Task Assignment with No Commitment, Full Coordination

Given:

$C_{n\in Z_s}^{i\in Z_n}$: The set of all drones in the mission area

$t_{n\exists Z_s}^{i\forall Z_n}(0)$: The initial flight times of all drones

r_n: The communications range of all drones

Ξ_n: The set of mission capabilities of the drones in the nth class

U_n: The set of physical capabilities of the drones in the nth class

$F_{n\exists Z_s}$: The dynamical model of the drone in the nth class

M: The set of all current and future tasks in the mission area

Do until $M = \emptyset \vee t_{n\exists Z_s}^{i\forall Z_n} = 0$:

    For each class n in C:

        For each agent i in C_n:

            CoordinatedTasking(i)

    EvolveTargets()

    EvolveAgents()

End Do

Algorithm 3 presents a viable approach to coordination between agents. In the CoordinatedTasking () function, agents will only choose tasks from the pool of tasks not being serviced by other agents. However, this approach ignores the communication requirements implied by coordinated tasking, in which agents will need relatively up-to-date information on their peers' tasking decisions.

Algorithm 4: Auctioning Task Assignment with Full Commitment, Full Coordination

Given:

$C_{n\in Z_s}^{i\in Z_n}$: The set of all drones in the mission area

$t_{n\exists Z_s}^{i\forall Z_n}(0)$: The initial flight times of all drones

r_n: The communications range of all drones

Ξ_n: The set of mission capabilities of the drones in the nth class

U_n: The set of physical capabilities of the drones in the nth class

$F_{n\exists Z_s}$: The dynamical model of the drone in the nth class

M: The set of all current and future tasks in the mission area

Do until $M = \emptyset \vee t_{n\exists Z_s}^{i\forall Z_n} = 0$:

    For each class n in C:
        For each agent i in C_n:
            If $i.task$ is None:
                CoordinatedTasking(i)
    EvolveTargets()
    EvolveAgents()
End Do

Algorithm 4 represents an alternative method of accomplishing coordinated tasking without the issue of agents suboptimally switching tasks. However, as in Algorithm 3, this approach does nothing to ensure that each agent is able to make its CoordinatedTasking() decisions using up-to-date information. Although the attempt to prevent agents from task switching will prevent suboptimal reassignment, it still carries a risk in an environment where task priorities may spontaneously change, requiring reconfiguration. This approach will serve as the baseline tasking approach used by the "Harvest" role of APIS autonomy, as the heuristic roles introduced in APIS will increase communication and mitigate the risks associated with task commitment, as agents will be better able to select more optimal tasks.

Algorithm 5: Auctioning Task Assignment with the APIS Framework and Behaviors

Given:

$C_{n\in Z_s}^{i\in Z_n}$: The set of all drones in the mission area

$t_{n\exists Z_s}^{i\forall Z_n}(0)$: The initial flight times of all drones

r_n: The communications range of all drones

Ξ_n: The set of mission capabilities of the drones in the nth class

U_n: The set of physical capabilities of the drones in the nth class

$F_{n\exists Z_s}$: The dynamical model of the drone in the nth class

M: The set of all current and future tasks in the mission area

Do until $M = \emptyset \vee t_{n\exists Z_s}^{i\forall Z_n} = 0$:

    For each class n in C:
        For each agent i in C_n:
            UpdateModelsFromComms(i)
            UpdateModelsFromObservation(i)
            UpdateModelsFromPrediction(i)
    EvolveTargets()
    EvolveAgents()

End Do

The main goal of APIS, at this stage, is to assign tasks based on internal models collected from communication, observations, and predictions. The self-task-assignment step is contained within the UpdateModelsFromPrediction() function. A key aspect of this approach is a tie-breaking preassigned ordering to model updates. In other words, two agents with identical sets of bids for identical tasks will not deadlock because the higher priority will assign itself. In contrast, the lower priority agent will predict the higher priority agent's assignment and act accordingly. Later, this assumption can be verified and corrected when communication is reestablished. We are able to implicitly avoid suboptimal retasking by ensuring that agents have an up-to-date model of their peers' tasking decisions and future states. We will focus our

analysis on the APIS autonomy framework; for a more detailed discussion on the relative merits of these presented methods, refer to [8].

Simulation Overview

To more thoroughly evaluate the potential for different APIS heuristic weightings to affect the swarm's overall performance, we created a new three-degree-of-freedom simulation capable of simulating multiple classes of tasks and multiple types of drones (each able to service different types of tasks). The simulation also randomizes the initial positions of both tasks and drones.

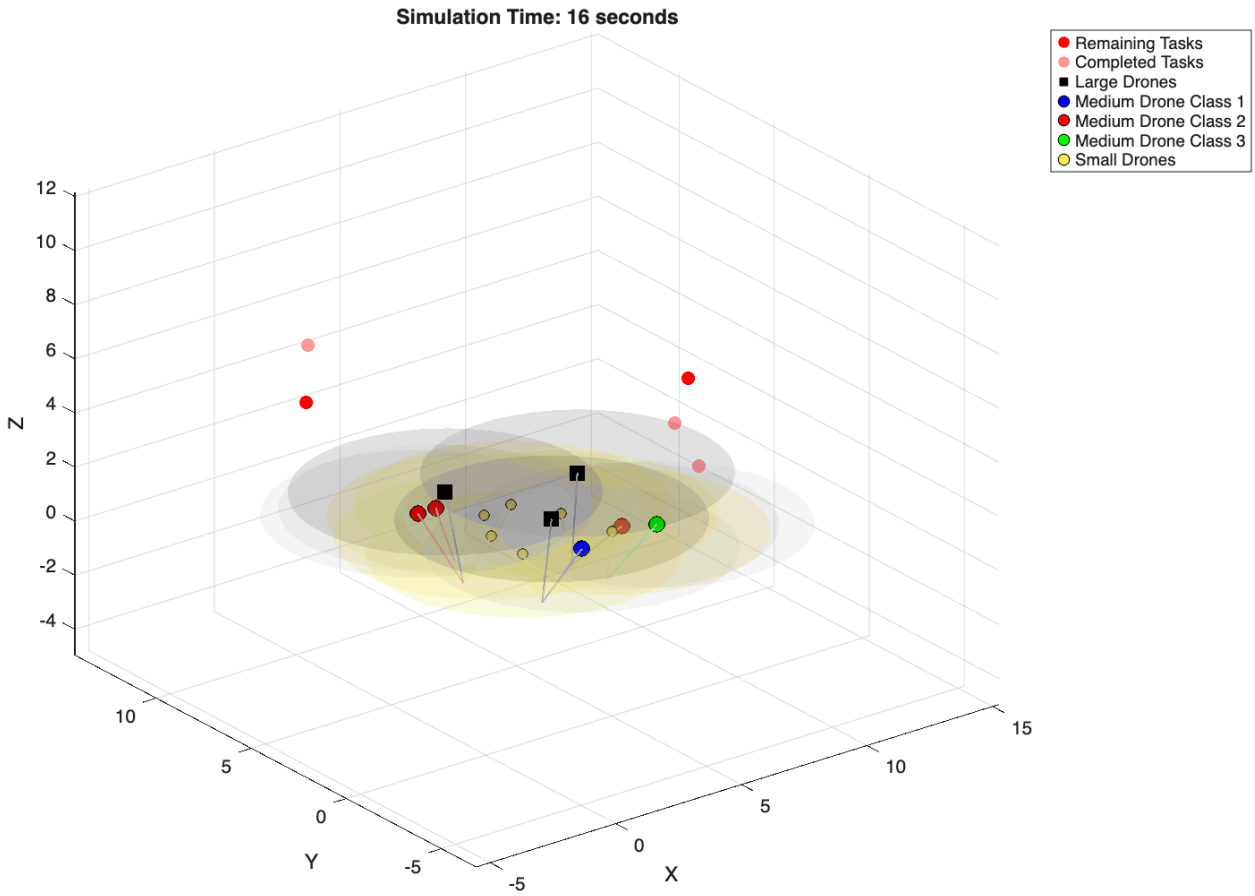


Figure 19: APIS simulation just after launching.

The simulation is semi-randomized, with agents starting randomly around the origin of the map and tasks uniformly distributed randomly in the area. This captures the trial-to-trial fluctuations of real-life flight testing and introduces the need to run multiple simulations to construct average scores.

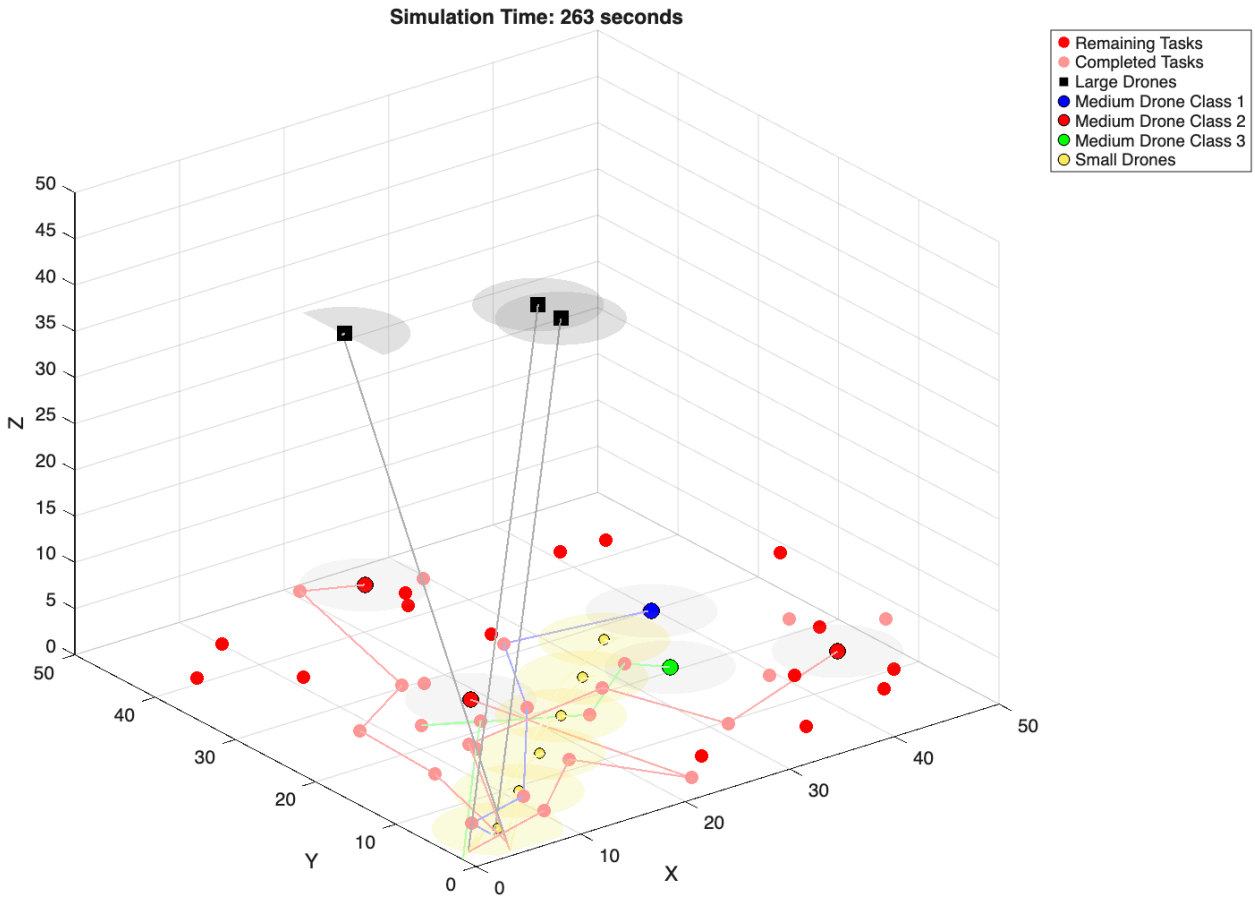


Figure 20: APIS simulation partway through the swarm operations.

After takeoff, the agents will embark on their first roles, randomly chosen from the task weightings of each class. The roles of each drone will be reassigned after milestones are met: for harvesting drones, after servicing a task; for scouting drones, after reaching a vertex of the environment; for dancing drones, after they reach their rendezvous point.

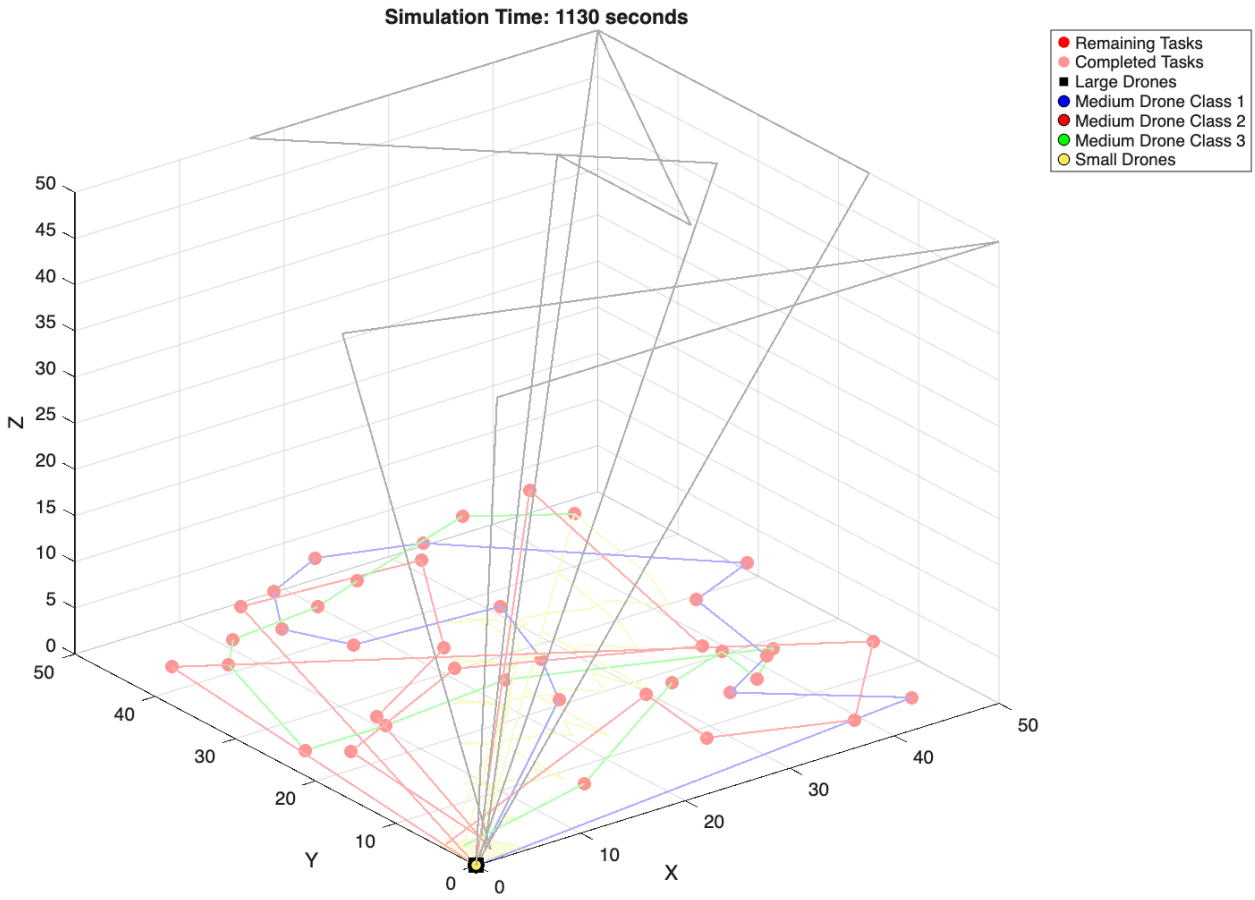


Figure 21: APIS simulation just after all drones return to the takeoff point.

Agents return to the takeoff point once all tasks have been completed. The concept of definite mission completion is a somewhat strong assumption but a necessary one in simulation to effectively calculate mission time-based scores.

We simulated the APIS algorithm in a three-dimensional simulation, with three types of tasks and three classes of agents. Each agent could complete either one, two, or all three types of tasks (“harvesting”). Two classes of agents could map the area (“scouting”), and all agents could serve as mesh nodes (“dancing”). Once all tasks are completed, the agents return to the takeoff area. We evaluated the swarm's performance using three criteria:

1. Scouting: The percentage of the operation area surveyed.
2. Harvesting: The ratio of the time the swarm took to complete all tasks compared to the ideal time with no communication limitations.
3. Dancing: Average connectivity, as measured by the average difference between each agent’s local models of other agents and those agents’ true states.

We varied the number of tasks, agents, and area sizes to evaluate how these performance indicators changed as the operational environment changed. We also varied the APIS role weightings for each scenario to adjust the frequency of agents taking on each role.

Table 2: APIS simulated performance in default configuration

Area Dimensions (m): 38x26x25 | 40 Tasks			**Type Definitions: [Mapping, Radio, Visual]** **Heavy Drone Types: [2, 0, 0]** **Medium Drone Types: [1, 1, 1]** **Light Drone Types: [0, 0, 3]**	
	Weightings: Large: [% harvesting, % scouting, %dancing] Medium: [% harvesting, % scouting, %dancing] Small: [% harvesting, % scouting, %dancing]			
Scores	[0, 1, 0] [1, 0, 0] [0, 0, 1]	[0.2, 0.8, 0] [1, 0, 0] [0, 0, 1]	[0.2, 0.8, 0] [0.9, 0.1, 0] [0, 0, 1]	[0.2, 0.8, 0] [0.9, 0.1, 0] [0.2, 0, 0.8]
Scouting	0.994	0.996	0.999	0.996
Harvesting	0.889	0.927	0.654	0.648
Dancing	0.812	0.804	0.837	0.779

In our default configuration, we notice two interesting trends: first, that the re-assignment of large drones from pure scouting to partial harvesting does not actually reduce scouting performance, while simultaneously increasing harvesting performance. This serves as an indication of “scouting saturation” - when the area is of sufficient size that it can be scouted completely, in less time than the system is deployed for. Second, we notice that the re-assignment of medium drones to partial scouting and light drones to partial harvesting negatively impacts harvest and dancing scores, respectively. This indicates a variable sensitivity of scores to our heuristic settings, particularly among the agents most responsible for those tasks.

Table 3: APIS simulated performance in heavy-task configuration

Area Dimensions (m): 38x26x25 | 100 Tasks			**Type Definitions: [Mapping, Radio, Visual]** **Heavy Drone Types: [2, 0, 0]** **Medium Drone Types: [1, 1, 1]** **Light Drone Types: [0, 0, 3]**	
	Weightings: Large: [% harvesting, % scouting, %dancing] Medium: [% harvesting, % scouting, %dancing] Small: [% harvesting, % scouting, %dancing]			
Scores	[0, 1, 0] [1, 0, 0] [0, 0, 1]	[0.2, 0.8, 0] [1, 0, 0] [0, 0, 1]	[0.2, 0.8, 0] [0.9, 0.1, 0] [0, 0, 1]	[0.2, 0.8, 0] [0.9, 0.1, 0] [0.2, 0, 0.8]
Scouting	0.999	0.999	1.000	1.000
Harvesting	0.326	0.920	0.609	0.601
Dancing	0.980	0.824	0.927	0.923

The second-configuration simulation results confirm the large impact of large drone heuristic tweaking on the potential to speed up task completion and increase harvesting score. We also notice that the “dancing” scores, measures of average model error, have actually improved over the default simulation case. This intuitively makes sense, as the increased number of tasks will lead to harvesting agents staying in closer contact with one another.

Table 4: APIS simulated performance in many-agents configuration

Area Dimensions (m): 38x26x25 | 40 Tasks			**Type Definitions: [Mapping, Radio, Visual]** **Heavy Drone Types: [3, 0, 0]** **Medium Drone Types: [1, 3, 1]** **Light Drone Types: [0, 0, 6]**	
	Weightings: Large: [% harvesting, % scouting, %dancing] Medium: [% harvesting, % scouting, %dancing] Small: [% harvesting, % scouting, %dancing]			
Scores	[0, 1, 0] [1, 0, 0] [0, 0, 1]	[0.2, 0.8, 0] [1, 0, 0] [0, 0, 1]	[0.2, 0.8, 0] [0.9, 0.1, 0] [0, 0, 1]	[0.2, 0.8, 0] [0.9, 0.1, 0] [0.2, 0, 0.8]
Scouting	0.997	0.997	1.000	1.000
Harvesting	0.667	0.674	0.544	0.698
Dancing	0.753	0.800	0.864	0.825

The many-agent simulation configuration demonstrates what may be considered the theoretical upper bound of APIS's usefulness. The harvesting score is calculated based on the ideal time, with no informational uncertainty, versus the actual time taken. In this scenario, the large number of agents deployed to the environment works against APIS, as the idealized simulation is able to dramatically outperform the more cautious APIS by aggressively deploying its large swarm. Our coordination protocols also work against our performance here, with more agents in total, meaning more agents are unable to select their first-choice task. This issue suggests that in order for APIS to be deployed at a larger scale, a more formalized distributed consensus algorithm may be called for.

Table 5: APIS simulated performance in large-area configuration

Area Dimensions (m): 76x52x50 | 40 Tasks			**Type Definitions: [Mapping, Radio, Visual]** **Heavy Drone Types: [2, 0, 0]** **Medium Drone Types: [1, 1, 1]** **Light Drone Types: [0, 0, 3]**	
	Weightings: Large: [% harvesting, % scouting, %dancing] Medium: [% harvesting, % scouting, %dancing] Small: [% harvesting, % scouting, %dancing]			
Scores	[0, 1, 0] [1, 0, 0] [0, 0, 1]	[0.2, 0.8, 0] [1, 0, 0] [0, 0, 1]	[0.2, 0.8, 0] [0.9, 0.1, 0] [0, 0, 1]	[0.2, 0.8, 0] [0.9, 0.1, 0] [0.2, 0, 0.8]
Scouting	0.988	0.966	0.984	0.974
Harvesting	0.846	0.868	0.702	0.705
Dancing	0.815	0.812	0.894	0.849

The large-environment trial mainly serves to demonstrate APIS's informational resiliency. Despite doubling the size of the area and significantly reducing the number of drones active (thus stretching the lines of communication), this configuration does not score considerably worse than our many-agents configuration and is only outperformed in the many-tasks case. This indicates that, as planned, our addition of the dancing heuristic task does present possible increases in connectivity over large areas.

Conclusions and Future Work

We have successfully demonstrated the capability of the APIS method of cooperative task assignment and heuristic behavior to reduce uncertainty, explore new environments, and maintain lines of communication between system agents. This has been demonstrated in simulation and is nearing completion in flight testing. However, the key demonstrator elements – that tasks can be delegated over an ad-hoc mesh network, that even large data structures such as LiDAR maps can be down-sampled and likewise exchanged, and that agents are able to serve as mesh network nodes – have already been validated. These markers demonstrate that the technological readiness (TRL) of APIS has increased and is nearing readiness for more large-scale field testing.

Our simulation trials have also provided insight into the impact of heuristic tuning on swarm performance. Different heuristic configurations performed better as we changed situational variables - number of agents, size of area, number of tasks. This indicates that a one-size-fits-all approach to APIS tuning is naive, and that a formalized tuning process should be developed in order to ensure that the APIS weightings are appropriate to the mission parameters presented. Further research should be conducted to characterize how changing mission specifications can interact with APIS heuristic weights and broaden the simulation environment to support different distributions of tasks, as well.

Our next steps are to continue validating APIS's performance in real-world flight tests. To further increase the TRL, we need to characterize the worst-case delays in the mesh network and the network reliability and continue to fine-tune the swarm's heuristic behavioral weightings. Further attention should also be given to the best employment of sensor payloads, as the basic functionality of an APIS swarm now opens the door to more complex inter-agent cooperative tasks.

Bibliography

[1] N. Atanasov *et al.*, “Decentralized active information acquisition: Theory and application to multi-robot SLAM,” *null*, 2015, doi: 10.1109/icra.2015.7139863.

[2] X. Cai, B. Schlotfeldt, K. Khosoussi, N. Atanasov, G. J. Pappas and J. P. How, "Energy-Aware, Collision-Free Information Gathering for Heterogeneous Robot Teams," in IEEE Transactions on Robotics, vol. 39, no. 4, pp. 2585-2602, Aug. 2023, doi: 10.1109/TRO.2023.3257512.

[3] X. Hu, Y. Xiong, Z. Zhang, and C. Li, “Consensus of a new multi-agent system via multi-task, multi-control mechanism and multi-consensus strategy,” *Neurocomputing*, 2024, doi: 10.1016/j.neucom.2024.127586.

[4] O. Manasherov and A. Degani, “Multi-Agent Target Allocation and Safe Trajectory Planning for Artificial Pollination Tasks,” *Smart Agricultural Technology*, 2024, doi: 10.1016/j.atech.2024.100461.

[5] E. Choi and W. Chang, “Multi-Agent Task Allocation with Interagent Distance Constraints,” *Journal of Aerospace Information Systems*, 2023, doi: 10.2514/1.i011272.

[6] K. Kondo *et al.*, “PUMA: Fully Decentralized Uncertainty-aware Multiagent Trajectory Planner with Real-time Image Segmentation-based Frame Alignment,” *IEEE International Conference on Robotics and Automation*, 2023, doi: 10.1109/icra57147.2024.10610629.

[7] Y. Chang, N. Hughes, A. Ray, and L. Carlone, “Hydra-Multi: Collaborative Online Construction of 3D Scene Graphs with Multi-Robot Teams,” *IEEE/RJS International Conference on Intelligent Robots and Systems*, 2023, doi: 10.1109/iros55552.2023.10341838.

[8] J. E. Pye and N. M. Alexandrov, “APIS: Honey Bee Foraging Task Assignment for Use in Uncertain and Unreliable Environments,” *AIAA SCITECH 2024 Forum*, 2024, doi: 10.2514/6.2024-0333.

[9] Corah, M., Michael, N. Distributed matroid-constrained submodular maximization for multi-robot exploration: theory and practice. *Auton Robot* **43**, 485–501 (2019). https://doi.org/10.1007/s10514-018-9778-6

[10] Z. Xu and V. Tzoumas, “Performance-Aware Self-Configurable Multi-Agent Networks: A Distributed Submodular Approach for Simultaneous Coordination and Network Design,” *null*, 2024, doi: null.

[11] A. Neumann, C. Aichele, M. Lindner, and S. Wunderlich, “Better approach to mobile ad-hoc networking (BATMAN),” *IETF draft*, pp. 1–24, Jan. 2008.

[12] V. Tripathi, L. Ballotta, L. Carlone, and E. Modiano, “Computation and Communication Co-Design for Real-Time Monitoring and Control in Multi-Agent Systems,” *International Symposium on Modeling and Optimization in Mobile, Ad-Hoc and Wireless Networks*, 2021, doi: 10.23919/wiopt52861.2021.9589966.

[13] Z. Xu, S. S. Garimella, and V. Tzoumas, *Communication- and Computation-Efficient Distributed Decision-Making in Multi-Robot Networks*. 2024. [Online]. Available: https://arxiv.org/abs/2407.10382

[14] M. Streeter, M. J. Streeter, D. Golovin, and D. Golovin, “An Online Algorithm for Maximizing Submodular Functions,” *null*, 2008, doi: null.

[15] Z. Xu, X. Lin, and V. Tzoumas, “Leveraging Untrustworthy Commands for Multi-Robot Coordination in Unpredictable Environments: A Bandit Submodular Maximization Approach,” *American Control Conference*, 2023, doi: 10.23919/acc60939.2024.10644567.

[16] Z. Xu and V. Tzoumas, “Resource-Aware Distributed Submodular Maximization: A Paradigm for Multi-Robot Decision-Making,” *IEEE Conference on Decision and Control*, 2022, doi: 10.1109/cdc51059.2022.9993308.

[17] L. Zhou, V. Tzoumas, G. Pappas, and P. Tokekar, “Distributed Attack-Robust Submodular Maximization for Multirobot Planning,” *IEEE Transactions on robotics*, 2022, doi: 10.1109/tro.2022.3161765.

[18] B. Schlotfeldt, V. Tzoumas and G. J. Pappas, "Resilient Active Information Acquisition With Teams of Robots," in IEEE Transactions on Robotics, vol. 38, no. 1, pp. 244-261, Feb. 2022, doi: 10.1109/TRO.2021.3082212.

[19] J. Wang and E. Olson, “AprilTag 2: Efficient and robust fiducial detection,” Oct. 2016.