APIS: Honeybee Foraging Task Assignment for Use in Uncertain and Unreliable Environments

John E. Pye* and Natalia M. Alexandrov†
NASA Langley Research Center, Hampton, Virginia 23681-2199, USA

Multiagent Cyber-Physical-Human (CPH) systems in realistic environments operate under uncertain conditions. Communication among agents, aimed at reducing the uncertainty, is itself subject to uncertainty. We propose to manage uncertainties in autonomous, long-duration operations of multiagent systems via a modified Honeybee Foraging (HBF) behavioral scheme. The resulting system, Autonomous Persistent Intelligent Swarm (APIS), incorporates two new behaviors to ameliorate informational uncertainty. When "scouting", agents are tasked based on informational quality and reliability rather than solely on priorities. When "dancing", agents are tasked to rendezvous with other dancing agents to exchange information at close range, where successful communication is guaranteed. When coupled with uncertainty-aware modeling across agents, these behaviors improve situational awareness and resilience of the system, enabling it to function under more uncertain conditions arising during long-duration missions.

I. The Problem

We propose a new method for cooperative, distributed control of a heterogeneous multiagent system operating under uncertainty. A Cyber-Physical-Human (CPH) team comprises interconnected systems—computers, physical systems governed by software, and humans—acting in a collaboration to accomplish a set of tasks [1]. A key prerequisite of successful CPH teaming is operational safety: humans must be able to work side-by-side with autonomous systems, without being exposed to significantly greater hazards than nominal. Hazards in CPH teams stem primarily from unpredictable, uncertain, or uncontrolled elements, which move the system from safe to unsafe states.

Operational safety in CPH teams has been extensively studied [2] [3] [4], with mitigation relying on predictability of the autonomous component behavior. Recent research into long-term safety strategies for CPH systems has suggested that such systems benefit from "anti-fragility", or the ability for the system to learn from mistakes and adapt to changing environments, as humans can [5]. Reliable operation of CPH teams depends on the system's robustness to uncertainty in data that inform the system's operation, as uncertain informational sources give rise to hazardous behavior. For example, swarms of airborne agents rely on an understanding of one another's states for avoidance of midair collisions [6].

In a multi-agent aerial system with many inspection and service targets, valuable ground equipment, and participating humans, there are many sources of uncertainty. To render such a system operationally viable, we require a method for coordinating the actions of agents in a way that minimizes uncertainty while maintaining efficiency. To manage uncertainties, we set out to incorporate direct management through model-based task assignment with adaptive, "anti-fragile" management, through model learning and improvement.

Here we focus on uncertainties due to the quality of communication among the system participants, as well as combined uncertainties arising from the limited accuracy and timeliness of the physical system state estimation, capabilities, decision-making of other agents, environmental perturbations, and mission-specific information on objectives, such as position and priority, among others. Understanding uncertainties and their propagation in agents' decision-making throughout the system informs task assignment and assessment of the outcomes' predictability and trustworthiness, which, in turn, inform the overarching decision-making of the human team members. Furthermore, such understanding allows for more predictable operation, reducing autonomous system performance variability and improving trust and CPH effectiveness. In this paper, we limit our investigation to the machine component of a CPH team, because machine trustworthiness is a prerequisite to that of the entire system.

Traditional task allocation in a swarm is based on task priorities. See [7] and [8] and references therein for an informative review of approaches. Task assignment algorithms vary significantly in effectiveness and applicability, from scenario to scenario. We examine a simplified scenario of servicing tasks in an informationally uncertain

*Pathways Student Research Engineer, Autonomous Integrated Systems Research Branch, AIAA Student Member.
†Principal Investigator, Autonomous Integrated Systems Research Branch, AIAA Associate Fellow

environment to create a basic platform for comparison of multiple fundamentally different task assignment algorithms.

Operations in uncertain environments are often addressed by introducing structure into the system. For example, drone swarms may fly in formation [9], or along preset paths to encourage or enable information sharing [10], or according to predefined rules of spacing and layout [11], or by maintaining constant communication between agents and ground control stations [12], or any number of other approaches aimed at instilling a degree of order and cooperation into a system, which otherwise may function in unpredictable ways. Adding structure, although effective at remedying specific scenarios, limits general adaptability of the system by potentially constraining an agent's autonomy (e.g., independence) too severely.

Biologically inspired autonomous systems strive to remedy this issue by focusing on the behavior of, and interactions among, agents over the course of a mission. As with general task assignment algorithms, research has been conducted on a wide variety of behavioral structures which exhibit swarming properties to solve numerical optimization problems, such as swarms of fish [13], fireflies [14], ants [15], with more biologically- and nature-inspired algorithms covered in [16], for further review. Although primarily examined in the context of general numerical optimization, the principles of such methods can be applied readily to physical problems of task assignment optimization, with similar benefits of effectiveness and reliability.

We propose to manage informational uncertainty via an analogue of Honeybee Foraging (HBF) [17] [18] behavioral approach. HBF, an extension of the Artificial Bee Colony [19] approach to numerical optimization, has shown promise in multi-objective optimization [20], a desirable trait when addressing the problem of many-to-many agent-task assignment. HBF relies on the scouting, foraging, and communicating behaviors exhibited by honeybees to find and share information about the problem space with other agents, and in so doing, generate consensus models of the environment, which may be used for optimization; in our case, for task assignment.

For the purposes of physical operations, we partition HBF behavior into three main "jobs": harvesting, scouting, and dancing. Honeybee harvesting functions similarly to standard auction-based algorithms for task assignment, wherein bees seek out food sources which they perceive to be of the highest value (characterized by high benefit and low cost for that bee). This standard harvesting behavior is augmented by scouting and dancing, which aim, respectively, to gather and disseminate information about the environment, to aid in task assignment.

Honeybee scouting is an exploratory behavior, where individual bees investigate the environment without prior information about the possible location of useful sites, such as sources of food or nesting locations. Honeybee dancing is a complex communication behavior occurring at preset sites (hives or colonies) which calls other bees to high-probability promising sites. We name the multiagent system controlled by our HBF scheme the "Autonomous Persistent Intelligent Swarm" or APIS. In addition to traditional priority-based task assignment, APIS employs two new, bee-inspired behaviors. To borrow honeybee terminology, we will refer to these behaviors as "scouting" and "dancing".

APIS scouting agents, as do harvesting agents, assign themselves to tasks not being done by other agents. However, rather than assign a task based on apparent priority (which might be incorrect, due to errors in information), scouting agents randomly assign themselves to a task with the objective of gathering information on that task. In doing so, scouts may collect information showing certain tasks to be of a higher or lower priority than is normally apparent to the swarm, thus adding adaptability to swarm task assignment. Without scouting, new information may not be collected, because harvesting agents act upon apparent task priority.

To aid in disseminating scouting information and information gathered by harvesting agents during the normal course of their mission, dancing agents rendezvous at a preset location for a given amount of time, with the sole goal of disseminating information. By choosing a rendezvous point located near high traffic corridors, dancing agents communicate their information to other agents passing by and to other dancing agents. The increased exchange of information further improves swarm coherency in environments with unreliable communication, by increasing the number of close-proximity encounters experienced by agents, potentially allowing for the use of shorter-range, more reliable methods of communication.

To facilitate taking advantage of disseminated information, all agents also maintain environmental models of other agents and tasks, which may then be updated using exchanged information or preset evolutionary models; for example, the inspection urgency of platforms in the environment growing over time at some set rate.

We hypothesize that this behavior will lead to improved resilience and system performance due to the enhanced exchange of higher-quality information. By gathering information on tasks which might not be investigated through scouting, by encouraging frequent exchange of information through dancing, and by storing information in a complete set of internal models, APIS agents will gain a more accurate picture of the tasking environment, in turn, leading to better task assignment.

The approach has the potential of retaining many of the optimality benefits of centralized coordination through informational consensus-building, while maintaining the resiliency and adaptability of a decentralized system. To test the validity of this hypothesis, we use a set of scenarios and behavioral parameters to compare the performance of APIS to a spectrum of auction algorithms presented in [21]. The comparison serves as a baseline test of the feasibility of APIS. Future experiments will include state-of-the-art algorithms to gain a more complete picture of the usefulness of APIS.

Here we report on a proof of concept, tested on a simplified multiagent system, across varying numbers of agents, targets, and various levels of uncertainty and job allotment. We observe and record the efficacy of the task assignment algorithm in reducing average task urgency and in reliability of performance across tasks. For each algorithm and scenario, we sum the priorities of targets in the mission area at each timestep and record an average priority score for the trial, with lower scores representing a comparatively more successful trial, with targets maintaining a lower priority. The trial scores are averaged across all trials to yield an average trial score metric, which determines the efficacy of the algorithm for a specific setting, relative to other algorithms we engage. Similarly, the standard deviation of the trial scores serves as a measurement of algorithm variability, with a higher standard deviation representing a larger expected variation in algorithm performance from trial to trial.

In the remainder of the paper, we state our assumptions, describe the algorithm in detail, describe the test case and the associated performance metrics, and conclude with preliminary computational results.

II. Assumptions and Problem Setting

In this paper, our Design Reference Mission (DRM) is inspection and potential repair over a set of target points in a space, such as open air, ground, outer space, or other environments where a multi-agent team moves. We formulate the DRM to be widely applicable to real-world problems, such as aerial inspection of degrading platforms over a difficult environment, such as a volume of space with sandstorms or dense fog.

Under realistic circumstances, task priorities and attendant informational uncertainties arise in a variety of forms and contexts. To maintain general applicability of the DRM, we compress these myriad priorities into a summary "urgency" metric, governing the servicing urgency of a particular task. We then apply a randomly generated distortion bias to that urgency, calculated from a standard deviation parameter of the simulation trial, governing the generation of distortion values across the simulation.

A. Targets

- Point (service) targets are static and randomly distributed over the inspection area.
- Only one target may occupy a given 1x1 grid square.
- A known baseline scalar urgency value is assigned to each target.
    - Baseline urgency is a given parameter, set across the entire trial and common to all targets.
- An unknown, randomly generated scalar distortion value is assigned to each target and added to its baseline urgency to form "true" urgency.
    - The distortion factor is hidden from non-servicing agents.
    - The distortion factor is constant over time.
- Urgency may vary with time.
    - For each time step where a target is not serviced by an agent, when a target and agent share a grid square, urgency increases by 1.
    - For each time step where a target is serviced by an agent, urgency decreases by 1 to a minimum of 0.
    - After remaining at 0 for a set amount of time, urgency resets to its base value.
- Priority evolution over time is known to agents.
- Evolution of apparent priority and true priority, from timestep to timestep is identical.
- Each target broadcasts its priority within a given radius. For this simulation, all transmission radii are equal.

B. Agents

- We make no *a priori* assumption on control scheme centralization. The optimal choice of strategy depends on the context of the operations and is one of the active subjects of this research effort. We hypothesize that hybrid or fully decentralized control scheme will be the prevailing scheme.
- For APIS agents, we assume full agent rationality; that is, given sufficient information, each agent is aware of the other agents' optimal strategy. This is a strong assumption for general agents operating in

complex systems. Given that the system under consideration here is a cooperative swarm with known objectives, the assumption is not unreasonable. However, communication failures can turn cooperative agents into non-cooperative ones, and this case represents one of the system's uncertainties.
    - In real-world environments, communication failure and resulting uncooperativeness could be the result of either communication equipment failure or of degraded environmental conditions, such as inclement weather.
- Agents prioritize reducing target urgency as efficiently as possible. A mathematical model of this objective function is shown in Section IV.
- Agents strive, when able, to avoid servicing the same target as other agents. Target servicing efficacy does not scale with the number of servicing agents at that target.
- Agent collision risk in the grid world simulation is negligible, due to grid size. Obstacle avoidance logic does not need to be considered during agent positional state evolution process.
    - In a real-world environment of platform inspection, collision avoidance would be handled on a lower level, by a guaranteeably safe controller.

C. Information and Sources of Uncertainty

- Priority broadcasts from targets may be biased based on distortion value.
    - APIS agents that have previously serviced a given target and know that target's true urgency can communicate the true priority to other agents.
- Priority broadcasts may not reach agents, either from targets or other agents.
    - Agent-to-agent and target-to-agent communication is limited by communication range. Agents outside of the communication range of other agents or targets are not able to receive communications sent from those sources.
- APIS agents maintain models of the position, current job, and servicing targets of other agents. APIS agents also maintain models of target urgency. All target positions and initial agent positions are known to all agents at the beginning of the mission.
    - APIS agents share models and broadcast information about themselves. Agents update their models when receiving first-hand information and when receiving information newer than their own model.
    - In the absence of external model updates, APIS agents evolve their agent model states according to the task assignment and control logic of the swarm. Agents also evolve their target models based on the modeled positions of agents which may be in servicing range.
- All agents share identical decision-making logic, rendering the collection of agents a true swarm.

III. Conceptual Approach

Given the set of all initial agent positions $\mathbf{X}_0^A$, all initial target positions $\mathbf{X}_0^T$, and all initial apparent (i.e., not compensated for unknown bias) target priorities $\mathbf{P}_0$, each agent i will choose some target j to visit and service, thus reducing target priority.

The objective of the swarm is to minimize the average level of priority in the mission area and to produce predictable servicing results, i.e., to minimize variation in average priority from instance to instance. The priority may take many shapes in real-world applications. For example, if the task is for agents to inspect degrading platforms in an area, the priority may represent the time since the platform was last inspected, with the objective to minimize that time across all platforms. Each agent operates based on an individual instance of the task assignment algorithm common to all agents.

To evaluate the efficacy of APIS, we compare its performance in completing the mission objective against the performance of a spectrum of baseline auction algorithms taken from [21], with varying levels of cooperation and target commitment. We use two performance measures, as follows.

At each timestep of every trial, we record the sum of current true priorities across all targets in the area. The sum is then averaged, yielding a total trial score. Across all trials of a particular algorithm under given conditions, we average all trial scores and compute the standard deviation of the trial score. The trial score average serves as a measure of effectiveness, tracking how well the algorithm matches agents to appropriate targets and generally provides service to the targets in the area. The standard deviation of the trial score measures the reliability (or lack thereof) of a particular algorithm, demonstrating how much the performance of a given algorithm is subject to change from trial to trial. This is an important metric in CPH teams, where autonomous system's predictability is a key criterion for trust and, thus, effective teamwork.

In the remainder of the section, we summarize the algorithms used in this study via pseudocode, to facilitate replicability. The term "bid", defined in Function 1 (Computational Algorithm section), refers to a value calculated based on a target priority and relative distance of an agent from the target.

Algorithm 1: Auctioning Task Assignment with Full Commitment, Full Coordination

Given:
A: Set of all agents in mission area
T: Set of all targets in mission area
B: Bounds of the mission area
u: Baseline urgency of all targets in mission area
σ: Standard deviation from baseline urgency of all targets in mission area
t: Timespan of mission
t_{reset}: Time before target priority reset
Initialize:
$t_i = 0$: Initialize current time
For each target j in **T**:
$x_j \leftarrow \mathrm{U}(\mathbf{B})$: Generate uniformly distributed, random, non-overlapping starting positional state of target j
$p_j \leftarrow u$: Initialize apparent priority of each target
$\Delta_j \leftarrow \mathrm{N}(u, \sigma)$: Generate normally distributed distorted urgency values for each target
$p_j^t \leftarrow \Delta_j$: Initialize true priority of each target
For each agent i in **A**:
$x_i \leftarrow \mathrm{U}(\mathbf{B})$: Generate uniformly distributed, random, non-overlapping starting positional state of agent i
Do until $t_i = t$
For each agent i in **A**:
If $\mathrm{Bid}(i, i.target) = 0$:
$i.target \leftarrow None$
For each agent i in **A**:
If $i.target = None$:
For each target j in **T**:
$i.bids[j] \leftarrow \mathrm{Bid}(i, j)$
For every other agent i_{other} in **A**:
If $\mathrm{InCommsRange}(i, i_{other})$:
$i.bids[i_{other}.target] \leftarrow None$
$i.target \leftarrow \max(i.bids)$
EvolveTargets(**T, A**)
For each agent i in **A**:
$x_i{}^+ \leftarrow ConvergeToTarget(x_i, i.target)$
End Do

Algorithm 1 represents the most cooperative version of the four auction algorithms presented for comparison. We hypothesize that its reliance on active communication to avoid double-servicing is likely to increase its variance slightly when compared to its fellow auction algorithms, due to the lack of guaranteed lines of communication in our scenario. However, this level of coordination is likely to yield some gains in efficiency, and its choice to commit to targets will avoid costly re-tasking. We thus conclude that Algorithm 1 will most likely outperform other auction variations, but may carry increased variance from trial to trial, reducing its usefulness in CPH teams.

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

Given:
A: Set of all agents in mission area
T: Set of all targets in mission area
B: Bounds of the mission area
u: Baseline urgency of all targets in mission area
σ: Standard deviation from baseline urgency of all targets in mission area

t: Timespan of mission
t_{reset}: Time before target priority reset
Initialize:
$t_i = 0$: Initialize current time
For each target j in **T**:
$x_j \leftarrow \mathrm{U}(\mathbf{B})$: Generate uniformly distributed, random, non-overlapping starting positional state of target j
$p_j \leftarrow u$: Initialize apparent priority of each target
$\Delta_j \leftarrow \mathrm{N}(u, \sigma)$: Generate normally distributed distorted urgency values for each target
$p_j^t \leftarrow \Delta_j$: Initialize true priority of each target
For each agent i in **A**:
$x_i \leftarrow \mathrm{U}(\mathbf{B})$: Generate uniformly distributed, random, non-overlapping starting positional state of agent i
Do until $t_i = t$
For each agent i in **A**:
If $\mathrm{bid}(i, i.target) = 0$:
$i.target \leftarrow None$
For each agent i in **A**:
If $i.target = None$:
For each target j in **T**:
$i.bids[j] \leftarrow \mathrm{Bid}(i, j)$
$i.target \leftarrow \max(i.bids)$
EvolveTargets(**T, A**)
For each agent i in **A**:
$x_i{}^+ \leftarrow ConvergeToTarget(x_i, i.target)$
End Do

Algorithm 2 retains the efficiency-improving commitment of Algorithm 1 but eschews coordination with fellow agents. Because coordination improves efficiency, Algorithm 2's performance is likely to be lower than that of Algorithm 1. However, in a limited-communication environment, the variable performance of cooperative algorithms should not affect the non-cooperative Algorithm 2. Thus, we project a slight improvement in reliability over Algorithm 1.

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

Given:
A: Set of all agents in mission area
T: Set of all targets in mission area
B: Bounds of the mission area
u: Baseline urgency of all targets in mission area
σ: Standard deviation from baseline urgency of all targets in mission area
t: Timespan of mission
t_{reset}: Time before target priority reset
Initialize:
$t_i = 0$: Initialize current time
For each target j in **T**:
$x_j \leftarrow \mathrm{U}(\mathbf{B})$: Generate uniformly distributed, random, non-overlapping starting positional state of target j
$p_j \leftarrow u$: Initialize apparent priority of each target
$\Delta_j \leftarrow \mathrm{N}(u, \sigma)$: Generate normally distributed distorted urgency values for each target
$p_j^t \leftarrow \Delta_j$: Initialize true priority of each target
For each agent i in **A**:
$x_i \leftarrow \mathrm{U}(\mathbf{B})$: Generate uniformly distributed, random, non-overlapping starting positional state of agent i
Do until $t_i = t$
For each agent i in **A**:
For each target j in **T**:
$i.bids[j] \leftarrow \mathrm{Bid}(i, j)$

For each other agent i_{other} in **A**:
If InCommsRange(i, i_{other}):
$i.bids[i_{other}.target] \leftarrow None$
$i.target \leftarrow$ max $(i.bids)$
EvolveTargets(**T, A**)
For each agent i in **A**:
$x_i^+ \leftarrow ConvergeToTarget(x_i, i.target)$
End Do

Algorithm 3 now discards the commitment of Algorithms 1 and 2 but retains the coordination of Algorithm 1. This approach is likely best suited in situations where task priority can vary wildly, which necessitates re-computation of task assignments, possible only in a noncommittal auction implementation. However, in our prosed DRM of gradually and predictably fluctuating priorities, such dramatic re-computation will likely hinder more than help, as agents re-target many times during their lifespan and waste travel time. Thus, we project Algorithm 3 to demonstrate worse performance than Algorithms 1 and 2. However, the strength and frequency of this noncommittal tasking may result in predictable (though sub-optimal) trials. Thus, Algorithm 3 is likely to exhibit reliability that is similar to that of Algorithm 2 and greater than that of Algorithm 1.

Algorithm 4: Auctioning Task Assignment with No Commitment, No Coordination

Given:
A: Set of all agents in mission area
T: Set of all targets in mission area
B: Bounds of the mission area
u: Baseline urgency of all targets in mission area
σ: Standard deviation from baseline urgency of all targets in mission area
t: Timespan of mission
t_{reset}: Time before target priority reset
Initialize:
$t_i = 0$: Initialize current time
For each target j in **T**:
$x_j \leftarrow$ U(**B**) : Generate uniformly distributed, random, non-overlapping starting positional state of target j
$p_j \leftarrow u$: Initialize apparent priority of each target
$\Delta_j \leftarrow$ N(u, σ) : Generate normally distributed distorted urgency values for each target
$p_j^t \leftarrow \Delta_j$: Initialize true priority of each target
For each agent i in **A**:
$x_i \leftarrow$ U(**B**) : Generate uniformly distributed, random, non-overlapping starting positional state of agent i
Do until $t_i = t$
For each agent i in **A**:
For each target j in **T**:
$i.bids[j] \leftarrow$ Bid(i, j)
$i.target \leftarrow$ max $(i.bids)$
EvolveTargets(**T, A**)
For each agent i in **A**:
$x_i^+ \leftarrow$ ConvergeToTarget$(x_i, i.target)$
End Do

Algorithm 4 is both greedy and noncooperative. Lack of coordination and commitment is likely to severely harm its mission efficacy. Furthermore, due to its total lack of cooperation with other agents, Algorithm 4 is most susceptible to changes in environmental layout and starting positions. Thus, we predict Algorithm 4 to have similar or worse reliability than Algorithm 1, and the worst performance out of the four.

Algorithm 5: APIS Task Assignment

Given:

A: Set of all agents in mission area
T: Set of all targets in mission area
B: Bounds of the mission area
u: Baseline urgency of all targets in mission area
σ: Standard deviation from baseline urgency of all targets in mission area
t: Timespan of mission
t_{reset}: Time before target priority reset

Initialize:
$t_i = 0$: Initialize current time
For each target j in **T**:
$x_j \leftarrow \mathrm{U}(\mathbf{B})$: Generate uniformly distributed, random, non-overlapping starting positional state of target j
$p_j \leftarrow u$: Initialize apparent priority of each target
For each agent i in **A**:
$x_i \leftarrow \mathrm{U}(\mathbf{B})$: Generate uniformly distributed, random, non-overlapping starting positional state of agent i
For each agent i in **A**:
$i.agentModels \leftarrow \mathbf{A}$
$i.targetModels \leftarrow \mathbf{T}$
For each target j in **T**
$\Delta_j \leftarrow \mathrm{N}(0, \sigma)$: Generate normally distributed distorted urgency values for each target
$p_j^t \leftarrow \Delta_j + u$: Initialize true priority of each target
Do until $t_i = t$
For each agent i in **A**:
For each other agent i_{other} in **A**:
ReconcileModels(i, i_{other})
EvolveTargetDiagnostics(**T, A**)
For each agent i in **A**:
For each target j in **T**:
UpdateTargetModel(i, j)
For each agent model i_{model} in $\{i, i.agentModels\}$:
UpdateAgentModel(i, i_{model}, t_i)
EvolveTargetReset(**T**)
For each agent i in **A**:
EvolveTargetReset$(i.targetModels)$
End Do

Algorithm 5, APIS, incorporates the strongest aspects of Algorithms 1-4. While APIS coordinates with other agents as do Algorithms 1 and 2, it resolves the largest weakness of coordination—communication disruption—by iteratively generating and improving upon models of the surrounding environment and fellow agents, for use during communication disruptions. This enables APIS to retain the gains in efficiency brought through coordination, while avoiding the pitfalls of increased variability due to communication breakdowns. Furthermore, the ability of APIS to record and disseminate distortion factors of targets in the mission space is likely to improve both the efficiency of the algorithm, as agents are more frequently tasked to targets in need of servicing, and the resistance of the system to distortions and disturbances. Finally, the model convergence based on model reconciliation logic is likely to reduce variability further, as agent performance becomes more dependent on mission runtime than on mission area layout.

IV. Computational Algorithm

Given the initial values of urgency, position of all targets, and locations of all agents, each agent i will calculate a bid for each target j based on its priority P and relative distance d. In our example of aerial agents surveilling and inspecting various platforms in an area, this is based on both the ease of inspecting a given platform (as a function of distance and other variables) and the need of that platform to be inspected.

Function 1: Bidding function Bid()
Agent i : argument 1
Target j : argument 2
$B_i^j \leftarrow P_i^j - d(x_i, x_j)$

return B_i^j
End Function

Where:

- P_i^j is the priority of target j as known by agent i
- $d(x_i, x_j)$ is the distance between target j and agent i

At every timestep t, each agent i will move towards position x of its goal target g. Agent movement ceases once it has converged on its target and recommences once a new target has been chosen. In our simulation, the area is represented by a grid, and agents converge according to Manhattan (straight line-segment) iterative paths. In a higher-fidelity simulation and in a real-world environment, this function would also include obstacle avoidance.

Function 2: State evolution function ConvergeToTarget()

Agent state x_i : argument 1
Target state x_{target} : argument 2
$\Delta = x_i - x_{target}$
If $\Delta.\mathrm{x} = \Delta.\mathrm{y}$ and $\Delta \neq 0$:
    If $\Delta.\mathrm{y} > 0$:
        ${x_i}^+.y \leftarrow x_i.y - 1$
    Else:
        ${x_i}^+.y \leftarrow x_i.y + 1$
Else If $\Delta.\mathrm{x} > \Delta.\mathrm{y}$:
    If $\Delta.\mathrm{x} > 0$:
        ${x_i}^+.x \leftarrow x_i.x - 1$
    Else:
        ${x_i}^+.x \leftarrow x_i.x + 1$
Else If $\Delta.\mathrm{x} < \Delta.\mathrm{y}$:
    If $\Delta.\mathrm{y} > 0$:
        ${x_i}^+.y \leftarrow x_i.y - 1$
    Else:
        ${x_i}^+.y \leftarrow x_i.y + 1$
$x_i \leftarrow {x_i}^+$
End Function

Where:

- x_i is the positional state of agent i
- x_{target} is the positional state of agent i's target

At every timestep *t*, the apparent and actual priorities of each target *j* will evolve according to whether it is being serviced by any agent *i*. Servicing agents, defined as agents in the same grid square as the target, will be made aware of the true priority and, thus, the distortion factor of the target. Non-servicing agents will only be made aware of the apparent priority of the target. This mimics realistic situations, as the condition of targets degrades and their relevant diagnostic signals change to reflect that.

Function 3: Priority evolution function EvolveTargets()

Set of all targets **T**: argument 1
Set of all agents **A**: argument 2
For each target j in **T**:
    $oldPriority \leftarrow P_j^t$
    $beingServiced \leftarrow False$
    For each agent i in **A**:
        If $x_i - x_j = \mathbf{0}$:
            $beingServiced \leftarrow True$
    If $beingServiced$**:**
        ${P_j}^+ \leftarrow \max(P_j - 1, 0)$

$P_j^{t+} \leftarrow \max(P_j^t - 1, 0)$

Else:

$P_j{}^{+} \leftarrow P_j + 1$

$P_j^{t+} \leftarrow P_j^t + 1$

If $oldPriority = P_j^{t+}$:

$j.resetTime \leftarrow j.resetTime + 1$

If $j.resetTime = t_{reset}$:

$P_j{}^{+} \leftarrow u$

$P_j^{t+} \leftarrow \Delta_j + u$

End Function

Where:

- P_j^t is the actual priority of target *j*
- P_j is the apparent priority of target *j*
- u is the baseline urgency assigned to all targets in the mission area
- Δ_j is the distortion bias applied to the urgency of target *j* to yield the true urgency

In APIS, Function 3 is broken up into two parts: first, each target's priority is updated. After agents evolve, the targets check for reset status.

Function 4: Priority evolution without reset function EvolveTargetDiagnostics()

Set of all targets **T**: argument 1

Set of all agents **A**: argument 2

For each target *j* in **T**:

$oldPriority \leftarrow P_j^t$

$beingServiced \leftarrow False$

For each agent i in **A**:

If $x_i - x_j = \mathbf{0}$:

$beingServiced \leftarrow True$

If $beingServiced$**:**

$P_j{}^{+} \leftarrow \max(P_j - 1, 0)$

$P_j^{t+} \leftarrow \max(P_j^t - 1, 0)$

Else:

$P_j{}^{+} \leftarrow P_j + 1$

$P_j^{t+} \leftarrow P_j^t + 1$

End Function

Function 5: Target reset function EvolveTargetReset()

Set of all targets **T**: argument 1

For each target *j* in **T**:

If $oldPriority = P_j^{t+}$:

$j.resetTime \leftarrow j.resetTime + 1$

If $j.resetTime = t_{reset}$:

$P_j{}^{+} \leftarrow u$

$P_j^{t+} \leftarrow \Delta_j + u$

End function

Where:

- $j.resetTime$ is the number of timesteps in a resettable state (i.e., with no priority change)

When two agents are within communications range, their APIS algorithms exchange information on respective models, with each model updating to form a consensus. In a real-world scenario, the reconciliation would occur

using each agent's onboard communication subsystems and may be subject to additional sources of error; for example, if the message is lost in transmission. For the purposes of simulation, the error is compressed simply into a communication range metric. However, future simulations with higher-fidelity scenarios will, in addition, simulate this uncertainty to demonstrate the capability of the algorithms under study to adapt to communication error.

Function 6: Reconcile sets of models function ReconcileModels()

Agent i : Argument 1
Other agent i_{other}: Argument 2
If $|x_i - x_{other}| > communicationRange$: Return
For each agent model m in $i.agentModels$:
    If $m.id = i_{other}.id$:
        $m_i \leftarrow i_{other}$
    Else:
        If $m_i.age > m_{other}.age$:
            $m_i \leftarrow m_{other}$
For each target model m in $i.targetModels$:
    If $\Delta_{m_{other}} \neq 0$:
        $\Delta_m \leftarrow \Delta_{m_{other}}$
    If $m_i.age > m_{other}.age$:
        $m_i \leftarrow m_{other}$
End Function

Where:

- $m.id$ is a unique identifying tag associated with m
- $i_{other}.id$ is a unique identifying tag associated with i_{other}
- $m_i.age$ is the time since m_i has been directly updated from the model source of truth, i.e., the modeled agent or modeled target
- Δ_m is the distortion factor accounted for in target model m, defined to be zero until the agent can retrieve its value from another agent or from the target itself during servicing.

The model governing each agent is updated by that agent's APIS algorithm, to maintain models of other agents and targets despite periods of communication breakdown. This persistence allows for asynchronous and distributed decision making in situations where agents are isolated or possess unreliable communication systems. Because all agents are controlled by identical decision-making algorithms, the model update logic applies to all agents' task assignment.

Each target in the mission space updates its own priority, communicates the updated apparent priority to all agents in range, and communicates its distortion factor to all servicing agents. This process simulates the constant flow of information about task targets to their servicing agents in the mission area.

Function 7: Evolve models of targets function UpdateTargetModel()

Agent i : argument 1
Target j : argument 2
If $|x_i - x_j| > communicationRange$:
    $i.m_j.age \leftarrow i.m_j.age + 1$
    EvolveTargets$(i.targetModels, i.agentModels)$
Else:
    $i.m_j.age \leftarrow 0$
    $i.m_j.resetTime \leftarrow j.resetTime$
    $P_{i.m_j} \leftarrow P_j.$
    If $|x_i - x_j| = 0$:
        $i.m_j.\Delta_j \leftarrow \Delta_j$
End Function

Each agent in the mission space is modeled by all other agents in APIS. Each agent updates its current job based on its current job completion status and random parameters. Agents then choose a new target to service. Finally, each agent updates its own state to proceed to its chosen target.

Function 8: Evolve models of agents function UpdateAgentModel()

Agent i : Argument 1
Agent model i_{model} : Argument 2
Current timestep t : Argument 3
$jobDone \leftarrow False$
If $t = 0$:
    $jobDone \leftarrow True$
    $i_{model}.currentJob \leftarrow Harvest$
    $i_{model}.target \leftarrow None$
Else:
    If $i_{model}.currentJob = Harvest$ OR $Scout$:
        $jobDone \leftarrow (P^{t}_{i_{model}.target} = 0)$
    Else If $i_{model}.currentJob = Dance$:
        If $i_{model}.danceTime = goalDanceTime$:
            $jobDone \leftarrow True$
            $i_{model}.danceTime \leftarrow 0$
        Else:
            $i_{model}.danceTime \leftarrow i_{model}.danceTime + 1$
    If $jobDone$:
        $i_{model}.currentJob \leftarrow$ choose($\{Harvest, Scout, Dance\}$)
        If $i_{model}.currentJob = Scout$:
            $i_{model}.target \leftarrow$ choose($i.freeTargetModels$)
        Else If $i_{model}.currentJob = Dance$:
            $i_{model}.target \leftarrow danceRallyPoint$
            $i_{model}.danceTime \leftarrow 0$
    If $i_{model}.currentJob = Harvest$:
        $switchTarget \leftarrow False$
        If $i_{model}.target = None$:
            $switchTarget \leftarrow True$
        Else:
            If ($i_{model}.target \in \{i.otherAgentModels.target\}$) OR ($P^{t}_{i_{model}.target} = 0$):
                $switchTarget \leftarrow True$
        If $switchTarget$:
            $i_{model}.target \leftarrow None$
            For each model m in $i.freeTargetModels$:
                $i_{model}.bids[m] =$ Bid(i_{model}, m)
            $i_{model}.target =$ max ($i_{model}.bids$)
        ConvergeToTarget($i_{model}, i_{model}.target$)
    $i \leftarrow i.agentModels[i.id]$
End Function

Where:

- $i.currentJob$ denotes the current job of agent i, which can take one of three values:
    - $Harvest$: The agent will seek out targets to service based on their priority.
    - $Scout$: The agent will randomly select a target.
    - $Dance$: The agent will congregate around a common rally point for a predefined stretch of time.
- $i.danceTime$ denotes the number of timesteps the agent has spent at the predetermined dancing rally point.
- choose() chooses one element from a set of elements, each with its own probability weight.

- $P^{t}_{i_{model}.target}$ denotes the modeled true priority of the target selected by i_{model}.
- $i.freeTargetModels$ is the set of all targets modeled by agent i not assigned to any agents modeled by agent i.

V. Preliminary Proof of Concept

For the purposes of comparison among algorithms, the DRM is the operation of a swarm of agents in an environment, performing a service. In our simulation, the agents perform service tasks by occupying the same cell as the target, and when not servicing, move according to their next job. When using auction algorithms, agents move towards the next task, but under APIS, agents may move towards a common rally point instead. This DRM is representative of a broad variety of missions. An immediate example is a fleet of flying inspection agents, managing the condition of deployed platforms or ground infrastructure, which may degrade over time. The concept is not limited to flying vehicles. For instance, it applies to swarms of ground and climbing robotic agents inspecting and maintaining facilities on Earth and in space and planetary environments.

In our tests, for each algorithm, we perform a series of 100 trials and record the sum of all targets' priorities at each timestep, and then average the priorities to yield a trial score. We calculate effectiveness as the average trial score across all trials and algorithm variability as the standard deviation of the trial score across all trials.

A. Computational Setup

We compare the performance of the auction algorithms 1 through 4 with that of APIS, Algorithm 5. The simplified simulation environment has the following characteristics:

- Simulation world: A 10 by 10 square grid of cells, with each cell representing a potential target position. Agents can move horizontally and vertically on the grid; distances are calculated via Manhattan Distance.
- Distribution: All targets and agents are randomly distributed throughout the simulation world, with each target or agent occupying a different cell.
- Agents: A variable number of agents, each governed by an instance of a shared algorithm.
- Targets: A variable number of static targets, with starting priority normally distributed around a common starting value.
- Timing: Each agent can move 1 square per time step. Each serviced target reduces its priority by 1 point per time step. Each non-serviced target increases its priority by 1 point per timestep.
- Priority: Each target priority is randomly distributed around a common score of 30, using standard deviation.

B. Computational Results and Analysis

To compare behavior across algorithms, we test each one under a range of conditions:

- Environmental variability: We vary the standard deviation used to generate random distortions for true target priorities from 0 (resulting in no difference between actual and apparent priority) and one third of the common priority of all platforms in the simulation area (resulting in significant differences between actual and apparent priorities). The purpose is to assess performance under informational uncertainty.
- Number of agents and targets: We vary the number of active agents and targets from a small group (2 agents and 4 targets) to a larger group (10 agents and 20 targets). This evaluates the capability of the algorithm to function cohesively in large groups, without wasteful double-tasking.
- APIS behavioral characteristics: APIS incorporates random chance in transitioning between jobs. An instance of a job concludes for harvesting and scouting when the target has been fully serviced. A job concludes for a dancing agent when the agent has remained at the rally point for the prescribed amount of time. At the conclusion of a job, the agent will transition to harvesting, scouting, or dancing, based on a weighted random chance. By changing the weights assigned to each job, we can the strengths and weaknesses of various behavioral strategies.

We compare: CC (Committed and Coordinated Auctioning), CN (Committed and Not Coordinated Auctioning), NC (Not Coordinated and Committed Auctioning), NN (Not Committed and Not Coordinated Auctioning), and APIS.

Table 1: Algorithm Effectiveness (Trial Priority Mean)
Standard APIS settings of 80% chance for harvest, 10% chance for scout, 10% chance for dance

Trial	CC	CN	NC	NN	APIS
$\sigma = 0$, 2 agents, 4 targets	1059.667	1477.988	2451.295	3545.324	520.066
$\sigma = 4$, 2 agents, 4 targets	1133.942	1646.819	2710.702	3884.058	550.199
$\sigma = 10$, 2 agents, 4 targets	1089.404	1530.468	2553.710	3701.541	535.719
$\sigma = 0$, 10 agents, 20 targets	6253.692	6972.121	11768.047	19732.141	3448.477
$\sigma = 4$, 10 agents, 20 targets	6214.143	6948.015	11719.990	19528.225	3471.483
$\sigma = 10$, 10 agents, 20 targets	6322.831	7059.089	11833.616	19724.153	3373.418

Table 2: Algorithm Variability (Standard Deviation)
Standard APIS settings of 80% chance for harvest, 10% chance for scout, 10% chance for dance

Trial	CC	CN	NC	NN	APIS
$\sigma = 0$, 2 agents, 4 targets	487.595	678.301	851.527	991.106	218.478
$\sigma = 4$, 2 agents, 4 targets	557.374	774.681	952.250	1101.224	248.056
$\sigma = 10$, 2 agents, 4 targets	565.056	809.849	1048.445	1193.280	234.622
$\sigma = 0$, 10 agents, 20 targets	1717.756	1735.272	1842.847	1977.284	859.093
$\sigma = 4$, 10 agents, 20 targets	1904.007	1861.644	1872.522	1937.558	874.616
$\sigma = 10$, 10 agents, 20 targets	2000.557	1956.513	1971.477	2063.461	933.994

Table 3: APIS Tuning Effectiveness (Trial Priority Mean)
Where H = Harvest % Chance, S = Scout % Chance, and D = Dance % Chance

Trial	80H, 10S, 10D	40H, 30S, 30D	100H	100S	80H, 20S	80H, 20D
$\sigma = 4$, 10 agents, 20 targets	3471.483	3730.266	3126.123	3474.689	3353.603	3176.460
$\sigma = 10$, 10 agents, 20 targets	3373.418	3769.991	3046.632	3430.168	3451.245	3251.597

Table 4: APIS Tuning Variability (Standard Deviation)
Where H = Harvest % Chance, S = Scout % Chance, and D = Dance % Chance

Trial	80H, 10S, 10D	40H, 30S, 30D	100H	100S	80H, 20S	80H, 20D
$\sigma = 4$, 10 agents, 20 targets	874.616	597.441	1069.471	435.743	994.257	848.346
$\sigma = 10$, 10 agents, 20 targets	933.994	1060.595	1086.235	411.931	924.536	859.197

From the results shown in Table 1 and Table 2, we observe that APIS with a standard (8-% harvest, 10% scout, 10% dance) behavioral set consistently attains approximately half of the Trial Priority Mean as the next best auction algorithm and attains approximately half of the standard deviation as the next best auction algorithm. Consequently, independent of the number of agents, targets, or level of environmental uncertainty, APIS is both more effective (with a lower Trial Priority Mean) and more reliable (with a lower Algorithm Variability) than the auction algorithms we tested it against.

From the data in Table 3 and Table 4, we observe the effect of varying APIS behavioral parameters on the performance of APIS in the DRM. Comparing across effectiveness levels, the standard blend of APIS behaviors (80% harvest, 10% scout, 10% dance) has one of the best effectiveness scores across both a small variability regime and a large variability regime, with a couple of notable exceptions. At 100% harvesting, APIS algorithm carries better effectiveness in both regimes, as expected, because harvesting is the primary method for reducing priority in the area, due to prioritization of high-priority targets. At 100% scouting, APIS algorithm appears to carry effectiveness scores very similar to the standard APIS algorithm. This is surprising given that scouting—a random task selection with model-based deconfliction of target assignment—is used primarily to gather new data about platforms, with a large discrepancy between reported and actual results.

Looking at the other job variants of 100% harvesting, we surmise that a contributing factor to the apparent competitiveness of these variants of APIS is the single-job aspect. This point to the need for further work in integrating different jobs in APIS, allowing the swarm to naturally function more cohesively.

Finally, we observe superiority over the standard behavior in the 80% harvest, 20% dance behavioral variation, in both uncertainty regimes. This is understandable: the preponderance of dancing agents, coupled with the crowded environment, dramatically increases the effectiveness of dancing agents in communicating information to other agents of the swarm. Comparing across variability, the 100% harvest behavioral variation demonstrated the largest variability in both regimes. Likewise, 100% scouting with its completely random (but deconflicted) task assignment showed by far the least variability in both regimes of uncertainty.

Our findings indicate a tentative promise of APIS in solving the many-agent, many-target task assignment problem in uncertain environments, due to the preliminary demonstrated increased effectiveness and reliability over a group of standard auction algorithms.

Examining APIS variant data, we also conclude that parametric tuning is key to mission success, both in the DRM and in real-life missions. Increased scouting and dancing generally appear to decrease algorithmic variability, while increased harvesting appears to increase algorithm effectiveness. However, depending on the scenario and environment, increased scouting can also increase effectiveness, while proper blends of harvesting, scouting, and dancing can simultaneously achieve improved reliability and effectiveness.

We deem these results generally applicable to evaluating the trends of APIS performance relative to other algorithms. However, the sensitivity of APIS to behavioral variations implies the need for careful tuning of the algorithm in various domains. Moreover, the comparison of APIS with other algorithms was done for our own implementation of the algorithms under comparison, rather than potential existing commercial software. Future definitive comparison of APIS will have to conducted vs. established implementations of alternatives.

VI. Concluding Remarks

In summary, we are cautiously optimistic that APIS represents a promising solution to the problem of coordinating swarms of agents in an uncertain environment, while engaged in CPH team operations. We base this optimism on the apparent improvement in both effectiveness and reliability (represented by a decrease in variability) of APIS over fundamental varieties of a standard auction algorithm for task assignment. We expect this approach to show the most significant improvements in effectiveness and reliability in situations where information is unreliable but verifiable, as in our DRM, where the agents were able to collect accurate corrections to target priorities during servicing. However, we would expect these gains in effectiveness and reliability to erode as environmental conditions improve and scouting and dancing behaviors—useful for alleviating informational uncertainty, rather than direct improvement of effectiveness—become superfluous. Moving forward, we plan to test APIS against more specialized and state-of-the-art task assignment algorithms to begin tuning its parameters and identifying its most promising use cases, as well as testing it in a higher fidelity simulated environment, to yield data more directly applicable to real-world missions, such as aerial drones inspecting platforms spread throughout an environment. Finally, we intend to formulate more DRM examples, to evaluate the level of tuning required between different DRMs to reach algorithmic optimality of performance.

Acknowledgments

The authors would like to thank NASA's ARMD/TACP/Convergent Aeronautics Solutions (CAS) project for supporting this work.

References

[1] S. K. Sowe, E. Simmon, K. Zettsu, F. de Vaulx and I. Bojanova, "Cyber-Physical-Human Systems: Putting People in the Loop," in IT Professional, vol. 18, no. 1, pp. 10-13, Jan.-Feb. 2016, doi: 10.1109/MITP.2016.14.

[2] N. Nikolakis, V. Maratos, S. Makris, "A cyber physical system (CPS) approach for safe human-robot collaboration in a shared workplace," in Robotics and Computer-Integrated Manufacturing, vol. 56, pp. 233-243, 2019, ISSN 0736-5845, https://doi.org/10.1016/j.rcim.2018.10.003.

[3] A. M. Madni, D. Erwin, A. Madni, E. Ordoukhanian, P. Pouya and S. Purohit, "Next Generation Adaptive Cyber Physical Human Systems" in Defense Technical Information Center, Sep. 2019, Technical Report SERC-2019-TR-013.

[4] N. Ali, M. Hussain and J. -E. Hong, "Analyzing Safety of Collaborative Cyber-Physical Systems Considering Variability," in IEEE Access, vol. 8, pp. 162701-162713, 2020, doi: 10.1109/ACCESS.2020.3021460.

[5] E. N. Ceesay, K. Myers and P. A. Watters, “Human-centered strategies for cyber-physical systems security,” in EAI Endorsed Transactions on Security and Safety, April 2018, doi: 10.4108/eai.15-5-2018.154773.

[6] Alexander, B.; *Aircraft Density and Midair Collisions*, Proceedings of the IEEE, 58(3):377-381, 1970.

[7] Hamza Chakraa, François Guérin, Edouard Leclercq, Dimitri Lefebvre, “Optimization techniques for Multi-Robot Task Allocation problems: Review on the state-of-the-art,” in Robotics and Autonomous Systems, vol. 168, 2023, ISSN 0921-8890, https://doi.org/10.1016/j.robot.2023.104492.

[8] Khamis, A., Hussein, A., Elmogy, A, “Multi-robot Task Allocation: A Review of the State-of-the-Art,” In: Koubâa, A., Martínez-de Dios, J. (eds) Cooperative Robots and Sensor Networks 2015. Studies in Computational Intelligence, vol 604. Springer, Cham. https://doi.org/10.1007/978-3-319-18299-5_2.

[9] Y. Mulgaonkar, G. Cross and V. Kumar, "Design of small, safe and robust quadrotor swarms," 2015 IEEE International Conference on Robotics and Automation (ICRA), Seattle, WA, USA, 2015, pp. 2208-2215, doi: 10.1109/ICRA.2015.7139491.

[10] A. Khan, E. Yanmaz and B. Rinner, "Information merging in multi-UAV cooperative search," 2014 IEEE International Conference on Robotics and Automation (ICRA), Hong Kong, China, 2014, pp. 3122-3129, doi: 10.1109/ICRA.2014.6907308.

[11] Jamie Wubben, Francisco Fabra, Carlos T. Calafate, Juan-Carlos Cano, Pietro Manzoni, “A novel resilient and reconfigurable swarm management scheme,” in Computer Networks, vol. 194, 2021, ISSN 1389-1286, https://doi.org/10.1016/j.comnet.2021.108119.

[12] Capitan, J., Merino, L. & Ollero, A., “Cooperative Decision-Making Under Uncertainties for Multi-Target Surveillance with Multiples UAVs,” in J Intell Robot Syst vol. 84, pp 371–386, 2016, https://doi.org/10.1007/s10846-015-0269-0.

[13] Pourpanah, F., Wang, R., Lim, C.P. et al., “A review of artificial fish swarm algorithms: recent advances and applications,” in Artif Intell Rev vol. 56, pp. 1867–1903, 2023, https://doi.org/10.1007/s10462-022-10214-4.

[14] Jun Li, Xiaoyu Wei, Bo Li, Zhigao Zeng, “A survey on firefly algorithms,” in Neurocomputing, vol. 500, 2022, pp. 662-678, ISSN 0925-2312, https://doi.org/10.1016/j.neucom.2022.05.100.

[15] M. Dorigo, M. Birattari, C. Blum, M. Clerc, T. Stützle, A. F. T. Winfield, “Ant Colony Optimization and Swarm Intelligence,” in Lecture Notes in Computer Science, https://doi.org/10.1007/978-3-540-87527-7.

[16] S. Patnaik, Xin-She Yang, K. Nakamatsu, “Nature-Inspired Computing and Optimization,” in Modeling and Optimization in Science and Technologies, https://doi.org/10.1007/978-3-319-50920-4.

[17] Tereshko, Valery & Loengarov, Andreas, “Collective Decision-Making in Honey Bee Foraging Dynamics,” in Computing Information Systems, vol. 9, 2004.

[18] Karaboga, Dervis. “An Idea Based on Honey Bee Swarm for Numerical Optimization,” in Technical Report - TR06, 2005, Technical Report, Erciyes University.

[19] Dervis Karaboga, Bahriye Akay, “A comparative study of Artificial Bee Colony algorithm,” in Applied Mathematics and Computation, vol. 214, Issue 1, 2009, pp. 108-132, ISSN 0096-3003, https://doi.org/10.1016/j.amc.2009.03.090.

[20] Yi Xiang, Yuren Zhou, “A dynamic multi-colony artificial bee colony algorithm for multi-objective optimization,” in Applied Soft Computing, vol. 35, 2015, pp. 766-785, ISSN 1568-4946, https://doi.org/10.1016/j.asoc.2015.06.033.

[21] Matarić, M.J., Sukhatme, G.S. & Østergaard, E.H, “Multi-Robot Task Allocation in Uncertain Environments,” in Autonomous Robots, vol. 14, pp. 255–263, 2003, https://doi.org/10.1023/A:1022291921717.