Optimal Control Techniques for Heterogeneous UAV Swarms

Sami Mian
ECE Department
University of Pittsburgh
Pittsburgh, PA, USA
sam415@pitt.edu

John Hill IV
ECE Department
University of Pittsburgh
Pittsburgh, PA, USA
jth84@pitt.edu

Zhi-Hong Mao
ECE Department
University of Pittsburgh
Pittsburgh, PA, USA
zhm4@pitt.edu

Abstract—**Heterogeneous Unmanned Aerial Vehicle (UAV) swarms offer unique opportunities for solving multi-robot missions, but also introduce novel implementation challenges. In our study, we develop Heterogeneous Decentralized Receding Horizon Control (HD-RHC) for swarm management in search & rescue missions. This new technique builds upon existing multi-agent UAV work, but adds the capacity to manage a fleet of heterogeneous, diverse robot platforms that are equipped for different mission capabilities. Through high-fidelity simulation (AirSim), we derive an optimal controller, develop a method to find optimal weights for a specific mission focus, and provide a path to physical system validation. We analyze the efficiency and performance of HD-RHC controller, and discuss different ways this new method can be integrated into mission-management scenarios.**

Index Terms—**Unmanned Aerial Vehicle (UAV), optimal control, swarm robotics, decentralized control, heterogeneous multi-agent systems, search and rescue, simulation**

I. Introduction

Unmanned aerial vehicles (UAVs) have seen widespread adoption in sensing tasks, especially those in unknown or dangerous environments, mainly due to their customizable payloads and high maneuverability. Recent research has been directed to focus on the deployment and use of multiple UAV agents, called swarms, to accomplish a joint goal. There are two primary limitations to deploying UAV swarms: First, many operational environments and missions require in-situ adaptability to mission events. While this is the primary motivation for deploying a swarm, the swarm's overall ability to handle situations is limited by the payload's sensors, power, tools, etc. that each UAV carries. Second, technology advances rapidly with business and cultural needs, thus swarm capabilities must evolve alongside the newer demands. However the costs incurred by comprehensive fleet upgrades are substantial. In both cases, utilizing a heterogeneous swarm using different types of UAVs is paramount to increasing the overall flexibility and cost of the technology. These swarms pose new challenges to control architectures, as applying a uniform architecture to the swarm is no longer applicable. Most research on these control problems focus on homogeneous systems, assuming group performance may be extrapolated based on uniform platform characteristics and dynamics. This is not the case for several missions, such as search & rescue [1], structural & materials analysis [2], and space exploration [3]. A variety of platforms and sensor payloads are used in order to obtain mission success, thus indicating a heterogeneous swarm as the optimal force. There is limited capability to manage a large fleet of varied platforms using a robust control framework with current market products. Our study applies an optimal control technique called Decentralized Receding Horizon Control (D-RHC) in context of a search and rescue mission. Though the technique itself has been implemented for swarm organization, incorporating the use of heterogeneous platforms poses novel challenges. While victim discovery can succeed with any agent detection, hazard discovery is specialized, requiring a multitude of different tools and sensor payloads. We therefore introduce a novel technique: Heterogeneous Decentralized Receding Horizon Control (HD-RHC). This new control technique considers an individual platform's capabilities and limitations, usually derived from sensor payload and power resources, and modifies the optimization functions used in D-RHC to factor these aspects in the swarm motion and mission planning.

II. Related Work

A. Swarm Robotics

A "swarm" of robots is a group of robot agents capable of working cooperatively to achieve a common goal or demonstrate a specific collective behavior [4]. Individual agents in a swarm tend to have limited capabilities in terms of computing, sensing, or actuation. In order for a robot swarm to accomplish different kinds of tasks, the swarm must be flexible, robust, and easily scalable [5]. Some examples of optimal uses for robot swarms include: using multiple agents to explore and map an unknown area quickly [6], and using multiple platforms to accomplish a common task, such as transporting a large object [7]. Multi-agent systems also support higher levels of robustness in regards to mission failures, since the use of multiple platforms allows for redundancy and error checking [6], [8], [9]. One of the main features of swarm robotics is location-based distributed computing. They are able to distribute tasks and workload among agents uniformly throughout a large physical area, allowing for increased spatial awareness as well as manipulation of a larger portion of the surrounding environment [10], [11]. This is well demonstrated in hazardous

978-1-7281-9825-5/20/$31.00 ©2020 IEEE

environments since swarm robots grant increased situational awareness and mission capabilities without the inclusion of human intervention. Extensive research has investigated the formation and uses of UAV swarms, including for search and rescue activities [12]–[16], pollution detection [17], [18], surveillance [6], damage detection [19], and autonomous security [20].

B. Receding Horizon Control

Receding Horizon Control (RHC) is an optimal control technique which repeatedly solves an optimization problem, determining the next action for a platform in a projected timeframe [21], known as the "receding horizon," the next time step during which an action must be determined. Like other model predictive control schemes, developing RHC requires first defining the objective, constraint, prediction method, and horizon [21]. They are ideal for iterative development and rapid simulation due to their efficient performance. RHC is used across the field, with a particular interest in swarm robotics over the past few years [22], [23].

RHC has been proven as a phenomenal control strategy for decentralized swarm systems. In particular, Decentralized Receding Horizon Control (D-RHC) [24] focuses on using predictive information about nearby swarm members and collective goals to determine and optimize the appropriate cost-based objective functions. The prediction is based on current robot state information including control model information of other robots in the system at each epoch.

Although RHC does not require much tuning, it is still challenging to design a decentralized version due to combining a set of differing goals, costs, and constrains all to form an effective optimization function [25]. Furthermore, in a complex swarm it is increasingly difficult to determine the most efficient combinations of costs that can remain static due to the dynamic nature of the operating environment. Henderson et al. proposed using cost adaptation to quantify these different variables easily [25]. The optimization objective is generated using a set of user-provided cost and constraint functions, and solved using simple heuristics. This allows for the adaptive agents to perform tasks while optimizing efficiency and safety [26].

Several prior works have focused on using RHC for homogenous systems. [27] utilizes RHC for taxi distribution in a crowded city environment, [28] applies evolution-based RHC for smooth group trajectory shaping, and [29] uses RHC for multi-platform tracking of RF signals. Closer to our work, [30] presents the Receding Horizon Planning framework, which handles task scheduling for heterogeneous systems for search and rescue work in a confined environment.

C. Simulation

Since it is difficult and time-consuming to tune controllers and UAV configurations physically, many multi-agent system designs relies on high-fidelity simulation. There are several simulation environments that are designed for testing and training UAV flight controllers. Some are equipped with the popular open-source autopilot system called ArduPilot [31]. The Gazebo simulation environment for Robot Operating System™ (ROS) has support for UAVs and is a free open-source simulator. Additionally, NASA provides several open-source UAV simulation environments, including the Independent Configurable Architecture for Reliable Operations of Unmanned Systems (ICAROUS) system which supports several libraries for geofencing, sense and avoid, and formal method verification for safety-critical applications [32].

Our work utilizes Microsoft's AirSim simulator [33], which has been used for numerous first-stage UAV research applications. AirSim is a high-fidelity physical simulator built using the Unreal Engine which allows modeling & evaluation in a variety of environments and flight conditions. Previous work has shown that controllers designed and tuned within AirSim are transferable to physical UAV platforms with minimal changes [34]. Though recent advances have been added in support of the Drone Racing Lab [35], we provide additional mechanisms to support swarm controls engineering. These include modeling of centralized & mesh communication networks, a framework to provide HD-RHC control to UAV platforms, and tools to tune & analyze HD-RHC performance.

III. Designing the HD-RHC Controller

We are extending RHC control for high-level motion planning, dictating which missions and locations each UAV platform is responsible for surveying. The swarm implements a mesh-network [36], where UAV platforms rely on their neighbors to communicate with the whole swarm.

A key contribution to this project is formulating objectives uniquely suited to specific members of the swarm without introducing the need to implement a separate controller for each different UAV. Specifically, every member of the swarm should be able to accomplish some common tasks in the mission, while members with targeted configurations will only be applicable to specific objectives. As an example implementation, we investigate a targeted surveillance mission in which the swarm must observe a given search area, and the area has specific regions of interest that require specialized sensory payloads to analyze. Figure 1 describes an example search area A with two classes of mission points.

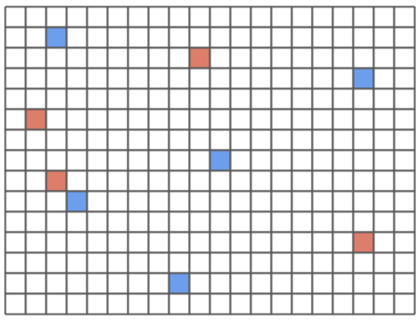


Fig. 1: Search area A with mission cells of two classes depicted in *red* and *blue*

A. Problem Concept

Suppose an operational area in which a heterogeneous UAV swarm executes a mission. This swarm is composed of UAV platforms, each with a payload (sensors, equipment, etc.) that makes it suited to accomplish a specified sub-mission. Cells that contain sub-mission objectives are called mission points. The swarm is provided with an *a priori* specification of the operational area and tasked with providing surveillance of the entire area, including specialized inspection of the mission points.

B. Parameter Definition

1) Payload Classes: The payload that each UAV carries provides that platform with unique capabilities and properties which may not be shared amongst the rest of the swarm. The variance of these payloads within the swarm is the basis of its heterogeneity. These payload classes are described by the set:

$$P : \{p^1, ..., p^k\} \tag{1}$$

Note that each UAV will have exactly one payload; we consider combinations of payloads to be a unique class in our problem, as it affects overall sub-mission acceptability and energy-costs.

2) Mission Points: The operational area contains some finite number of mission points, however these areas of interest are best assessed with specialized payloads. For our problem, we consider that each mission point has a mission uniquely satisfied by exactly one class of payload.

As such we define the set of mission points:

$$G : \{g_1, ..., g_m\} \tag{2}$$

Since the mission point is only applicable to a specific payload, we will use a superscript to denote its applicability with respect to a payload class. Hence, the major mission can be described as

$$x \mid g_j^x \in G \to p^x \in P \tag{3}$$

3) UAVs Individually: Each agent in the swarm has properties shared by the entire swarm. Particularly, we assume that any region without a mission point is capable of being surveyed by any member in the swarm. While this seems intuitive, it is important to recognize that payloads do not make UAV platforms orthogonally unique to one another – otherwise the problem is reducible to operating concurrent swarms.

Nonetheless, other non-mission properties are evident. These include pose (r) and velocity ($\dot{r}$) with respect to the origin of A and the energy-cost function (e) of UAV operations. The action of a UAV, $u_i(t)$, is idealized to the transit activity moving from one place to another. This includes the acceleration to achieve target flight velocity, deceleration to reach the location of interest, and surveillance maneuvers needed to be effective with the payload. The energy-cost function, therefore, is a function of both the activity and underlying properties of the payload (mass & its own energy needs during usage), hence its definition as $e(u(t); p)$.

An additional property is considered with UAV operation: reliability (ζ). This property represents the overall health of the individual UAV platform. For our problem, it does not matter if this property is formulated as a probability or a score. What we do consider, however, is that this property diminishes over the lifetime of operation. That is to say, reliability has the following dynamics:

$$\arg\max_t \zeta(t) = 0 \tag{4}$$

$$\lim_{t\to\infty} \zeta(t) = 0 \tag{5}$$

$$\frac{d\zeta}{dt} < 0, 0 \le t < \infty \tag{6}$$

With payload, we succinctly describe a UAV in swarm D as follows:

$$d_i \in D : \{p^x \in P, r_i, \dot{r}_i, e(u; p), \zeta\} \tag{7}$$

Each platform has a membership function as a consequence of sub-mission applicability. This function is idealized as:

$$m(g_k^x) = \begin{cases} 1 & p^x \in d_i \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

This concept is crucial when our problem considers both high-level (mission) and low-level (motion) planning objectives considering the payload-classes of members of the swarm.

C. Mission Planning Problem Formulation

The controller splits between high-level objective planning and low-level motion planning. High-level planning primarily focuses on the swarm satisfying the objectives stated by the mission points. A bidding scheme executed in each planning epoch is defined by the following function:

$$b_{i,j}(d_i, g_j) = \begin{cases} \|r_i - g_j^x\| - \zeta & m(g_j^x) > 0, g_j^x \in G^* \\ \infty & \text{otherwise} \end{cases} \tag{9}$$

Here, we define $G^* \subseteq G$ as the set of mission points that have not been visited by a UAV with an appropriate payload requirement matching d_i's payload. A mission point is won by having the lowest bid. UAVs can only have a single mission point during a planning epoch. So when a UAV is able to win multiple mission points, it will select the mission point that generated the lowest bid (greedy approach). In the event of a tie, a random-draw is negotiated by the UAVs. Bidding in this scheme is exercised at each planning epoch. This helps ensure robustness in the event that when a platform fails, another appropriate one can take its place.

In our experiments, we satisfy conditions (4), (5), and (6) for reliability by reducing ζ for each action taken and when a UAV visits a payload-applicable mission point. The additional reduction characterizes the reduced reliability of the UAV itself because its specialized payload was energized.

There may arise cases where a UAV simply does not win bidding on any mission point, such as when there are more platforms available than mission points. In that event, the motion planning function will drive the UAV to explore the operational area, maximizing search coverage. Likewise, we expect cases where $G^* = \emptyset$, meaning all mission points have been visited. We define the time of this event T_g.

D. Motion Planning Problem Formulation

1) Search Coverage Cost: Search coverage itself is a function of the swarm providing a "sweep" of the entire operational area. That is, we consider that UAVs may or may not have observability into neighboring cells with their default sensor suite. We define $v_i(r)$ as the region in the operational area which is being observed by UAV d_i. Therefore, the total area searched over time by the swarm can be defined as

$$a_i(t) = \int_0^t v_i(r(t))dt \tag{10}$$

Therefore we can consider a search cost c_a:

$$c_a = -v_i(r_i(t); u(t)) \tag{11}$$

Here, $u(t) \in U$ represents an action in the action space of the UAV. In our example case, we consider the function $v_i(r; u)$ as an update to a one-hot heat map describing the operational area with newly observed cells incrementing by 1, and repeated observed cells having no change. Therefore actions that lead to greater newly discovered cells have lower costs than those which do not. One consequence of describing the covered search area is that we can detect a time T_a such that the entire operational area A is covered. That is,

$$\lim_{t \to T_a} \sum_{i=1}^{N} a_i(t) = A \tag{12}$$

2) Mission Point Visualization Cost: Considering mission point selection, we consider the definition of error-distance to the mission point as the cost:

$$c_g = \begin{cases} \|r - \tilde{g}\| & \tilde{g} \in G^* \\ 0 & \text{otherwise} \end{cases} \tag{13}$$

Two results can be observed with this formulation. First, as a UAV with appropriate mission payload reaches the current mission point, its cost will increase so that it can move onto the next unvisited mission point whose bid it won. The second is that the situation when the cost reaches zero is only when there are no more appropriate mission points that the UAV should visit. Note that we define *time of mission complete* T_g as the time when $G^* = \emptyset$.

3) Energy Cost: Overall energy costs are important for any UAV's operation, They are critical when considering reusability and the likelihood of mission success – no one wants a UAV to fail before it completes its specified mission. Intuitively, a number of factors play into this cost, specifically its payload and the amount of force required for it to transit from one location to another. Here, we idealize the energy function of a given activity as $e(u; p)$:

$$c_e = e(u(t); p) \tag{14}$$

E. Full Optimization Problem

We characterize the total cost to be optimized as the weighted-sum of the aforementioned costs with weights w_a, w_g, and w_e respectively. Thus the swarm action process is defined as the solution to this optimization problem:

$$\begin{aligned} &\min_{u \in U} w_a c_a + w_g c_g + w_e c_e \\ \text{s.t..} &\|r_i - r_j\| > \delta_{min}, \quad \forall d_{i \neq j} \in D \\ &\lambda_{i,j} < \infty, \quad \forall d_{i \neq j} \in D \end{aligned} \tag{15}$$

We consider two constraining factors for the swarm: **safety** to keep UAVs from inadvertently colliding with one another; and **cohesion** to keep the swarm from separating too far so that mesh-network communication is impossible. These are competing constraints with respect to distance.

1) Safety Constraint: For our problem, we consider a safety-radius of δ_{min} from which each UAV must stand-off from one another. The constraint itself does not need to be uniform. In later work, we consider such cases where directional active-sensors need to be oriented away from each other to reduce interference during mission operations.

2) Cohesion Constraint: Unlike safety, cohesion cannot be considered as keeping within a maximum radius. Rather, consider that the swarm implements a *mesh network* [36]. We, therefore, must redefine this property. A swarm is cohesive when the swarm has an unbroken mesh network allowing any UAV to be in communications with any other UAV.

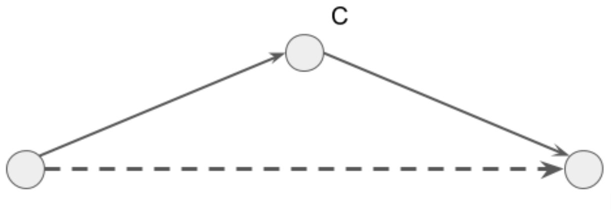


Fig. 2: Node Reachability Example: $\|A - B\| > \delta_{max}$; B is reachable because $\|A - C\| < \delta_{max}$ and $\|C - B\| < \delta_{max}$ and therefore $\lambda_{a,b}$ is finite.

As such, we consider the swarm itself as a weighted K-graph with weights, $\mathcal{L}_{i,j}$, between the nodes as:

$$\mathcal{L}_{i,j} = \begin{cases} \|r_i - r_j\| & \|r_i - r_j\| < \delta_{max} \\ \infty & \text{otherwise} \end{cases} \tag{16}$$

Here, δ_{max} represents the maximum range in which point-to-point communications is considered successful, beyond which, we consider the cost to be infinite. Following this, a number of path search techniques can be employed to provide a minimal-cost path for a message from any one UAV to reach with path-length $\lambda_{i,j}$. When there are no reachable paths between UAVs d_i and d_j, we define $\lambda_{i,j} = \infty$, hence the constraint for the swarm:

$$\lambda_{i,j} < \infty, \quad \forall d_{i \neq j} \in D \tag{17}$$

F. Cost Adaptation

As found in [25], this formulation allows for cost-adaptation, which is the tunability of the controller given characteristics of the swarm itself. To achieve this, we search for tuning that minimizes time to mission completion. Here, we define time to mission complete as:

$$T = \max(T_a, T_g) \tag{18}$$

Therefore, we tune the tuple (w_a, w_g, w_e) to minimize T.

IV. System Design

We provide a simulation setup to tune and evaluate the controller, demonstrating its scalability. This system consists of a client-end model of each UAV of the swarm and the mesh network model, which can connect to an optional server-end AirSim instance. A single discrete controller was provided to all platforms of the swarm, demonstrating its extensibility to a heterogeneous composition. This is made available to the public for inspection and testing.

A. Discrete Controller Design

In general, the controller has three high-level functions, described in Algorithm 1. At each planning epoch, each UAV accumulates the latest telemetry and bid information of the swarm, assesses its bid for mission points, and determines its next action to take.

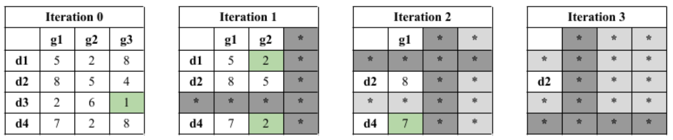
Algorithm 1 Discrete Controller Loop

```
Input: $t; H : \{(r_i(t), \zeta_i(t))\} \forall i = 0..N-1$
Output: $u \in U$
1: updateSwarmHistory($H$)
2: g = selectMissionPoint()
3: $u$ = computeMotionVector($g; H$)
4: return $u$
```

Each UAV uses the telemetry and bid information to update its understanding of the internal mission state. That is, it updates its understanding of search coverage and mission point visitation in addition to the location of each UAV in the swarm to understand constraint satisfaction.

B. Mission Point Selection

Mission point selection is assigned based upon the bidding scheme discussed in the previous section. However, each UAV can assess and determine the entire swarm's selection, given the information passed at each step through the mesh network. This is done by constructing a table of bids and allowing assignment to go in order of minimum bid.



Iteration 0			
	g1	g2	g3
d1	5	2	8
d2	8	5	4
d3	2	6	1
d4	7	2	8

Iteration 1			
	g1	g2	*
d1	5	2	*
d2	8	5	*
*	*	*	*
d4	7	2	*

Iteration 2			
	g1	*	*
*	*	*	*
d2	8	*	*
*	*	*	*
d4	7	*	*

Iteration 3			
	*	*	*
*	*	*	*
d2	*	*	*
*	*	*	*
*	*	*	*

Fig. 3: Example of mission point assignments

Bid tables comprised of elements $b_{i,j}$ computed by Equation (9) is constructed for each mission point class. Searching in order of minimum bid, rows and columns are eliminated when an assignment is selected. By going in order of minimum bid, each UAV is able to select mission points taking the greedy approach, and ties are resolved by simply looking up the pre-generated random rolls each bid makes for itself.

Using the example described by Figure 3 above, Iteration 0 describes an initial composition of bids. UAV d_3 has the minimum bid overall in the table, and is awarded g_3. Thus for Iteration 1, the row denoting d_3 and column denoting g_3 are eliminated, and the loop continues. Here, both UAVs d_1 and d_4 provide the lowest winning bids for g_2; in this scenario, d_1's random-draw was a lesser value than that of d_4, so it is awarded g_2. Iteration 2 demonstrates that though d_4 did not win the bid of g_2, it instead wins g_1. In Iteration 3, with all mission point columns eliminated, d_2 is left without assignment and thus favors search when evaluating action cost. Note that assignment also ends in the case where there are no UAVs left in the table, yet mission points remain.

C. Action Selection

With a mission point assignment, we can now consider potential actions. For a proof-of-concept, we explore a discretized grid map and provide an action-space such that each UAV selects which neighboring cell to visit (four cardinal directions and four diagonal directions). To keep the computational load small for this study, the controller does not consider potential next-states of the swarm. This is done so that the action space is not **size 8N** where **N** is the size of the swarm. Future work will consider reducing this space.

Algorithm 2 Motion Vector Selection

```
Input: $g \in G^*; H : \{(r_i(t), \zeta_i(t))\} \forall i = 0..N-1$
Output: $u \in U$
    Initialisation : $U^* = \emptyset$
1: for $u \in U$ do
2:     if $\|(r_i(t) + u) - r_j(t)\| \geq \delta_{min}$ and $\lambda_{i,j} < \infty \; \forall j \neq i$ then
3:         Add $u$ to $U^*$
4:     end if
5: end for
6: if $U^* \neq \emptyset$ then
7:     $u^* = \arg\min_{u \in U^*} w_a c_a + w_g c_g + w_e c_e$
8: else
9:     $u^*$ set to null-action
10: end if
11: return $u^*$
```

This example implementation first filters for actions that do not violate the optimization constraints, then searches for the action with minimal cost. In the event that every action violates the constraints, the UAV remains stationary (the "null-action") for its safety; this is necessary for the case where the entire swarm begins from a common staging location.

D. Controller Tuning

The weights w_a, w_g and w_e are calibrated to strike a balance between seeking to cover the search area and favoring mission point visitation. To do so, we consider two different techniques: Monte-Carlo (MC) and Simulated Annealing (SA) inspired by [24]. Both are uninformed search techniques that are commonly applied to such problems. As such, several trial iterations with different weight settings are analyzed to find the optimal combination, which minimizes time to mission complete (see Equation (18)).

V. Experiments & Analysis

Here, we execute two experiments to evaluate the controller: Benchmarking Tuning Effort, and Assessing Scalability of Tuning. In all cases, we are predominantly interested in three aspects: 1) Mission point visitation, 2) Search Area Coverage, and 3) Efficiency of Search Coverage. While the first two are parts of the controller's cost function, the third is interested in understanding how frequently cells in the space are revisited.

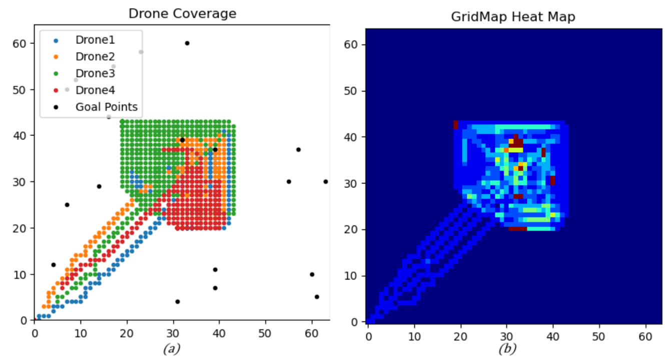


Fig. 4: Example mission coverage and efficiency plots regarding a failed configuration.

To tune, a search area of 64x64 is proposed with two different payload types. There are four UAVs whose positions were chosen arbitrarily near the origin. There are 20 mission points split evenly among the payload types. Our preliminary modeling was integrated with AirSim for accuracy and visual analysis.

Through the remainder of the document, we will review both mission coverage results via odometry and the efficiency of the swarm. Figure 3 provides an example: The coverage plots (a) provide an assessment of mission point visitation and total search area coverage; mission points (in black) which are not visited have no odometry paths joining them. Overall coverage can be understood by observing the odometry data spanning the graph field. The heat map (b) represents cells that have been repeatedly visited with hot/red, and cool/blue represents rarely or unvisited cells. This offers an effective understanding of loitering behavior.

A. Controller Tuning

Cost-adaptation is a key feature of D-RHC, and so it too is required of HD-RHC. There are several ways this controller can be tuned. However, we chose to explore two uninformed search techniques, Simulated Annealing (SA) and Monte-Carlo Method (MC).

In these setups, mission points are randomized between each simulation trial; UAV starting locations were not due to simulator configuration requirements. In all cases, it is expected that poor cost-weights may be analyzed, and so the simulations are capped with a maximum step-count.

1) Simulated Annealing: Simulated Annealing is a randomized search that attempts to explore local neighborhoods searching for optimal solutions [37]. This tuning method was chosen as it is a popular default heuristic search to locate local-optima and explore sub-optimal regions about it. It is, however, a non-deterministic search, so even when optimal configurations are determined after the set, it is recommended that multiple setups are executed and results compared with each other.

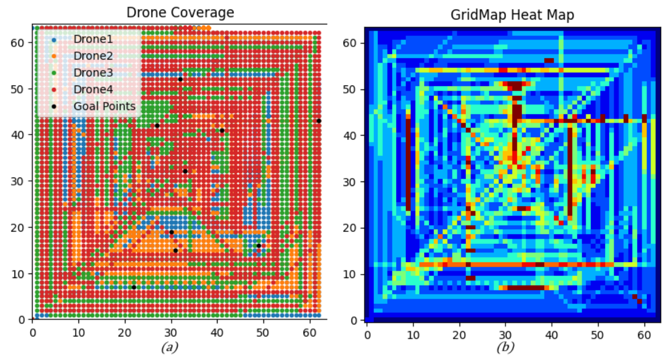


Fig. 5: Example results from SA Tuning.

In our implementation, we restricted the search space to only the positive-domain as negative weights would create a repulsion-effect in the controller (for example, if $w_g < 0$, then the UAV would favor going away from its assigned mission point). The search itself began at (1.0, 1.0, 1.0), randomly searching the neighborhood with a maximum difference magnitude of 1.0 from its currently selected "best" candidate configuration.

2) Monte-Carlo Method: Monte-Carlo Method is another uninformed search technique in that the search randomly polls about a neighborhood given a probability distribution. Readers wishing to explore implementations can refer to [38] and [39]. This method was also chosen because of its popularity and simplicity to implement. We restricted the search space to weights $0 < w \leq 1$ with uniform distribution.

Figures 6, 7, and 8 show a selection of results from the trial runs, where we look at the dynamics of each of the components of the cost function (by setting the other two weights to 0). Figure 9 shows the results of one of the highest performing trials.

B. Scalability Experiments

We also analyze the applicability of tuning parameters to larger mission areas and swarms. In tuning, the swarm itself was a four UAV configuration with only two payload classes.

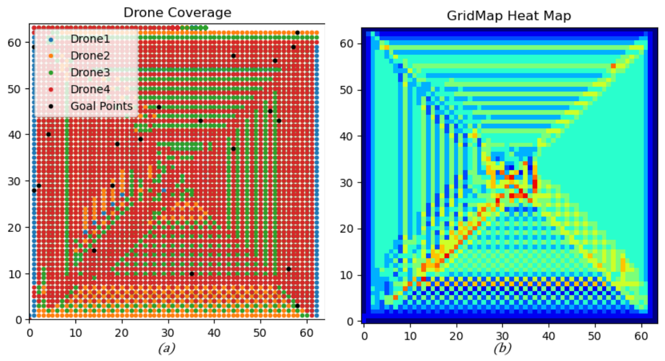


Fig. 6: MC trial with weights $w_a = 1$, $w_g = 0$, $w_e = 0$

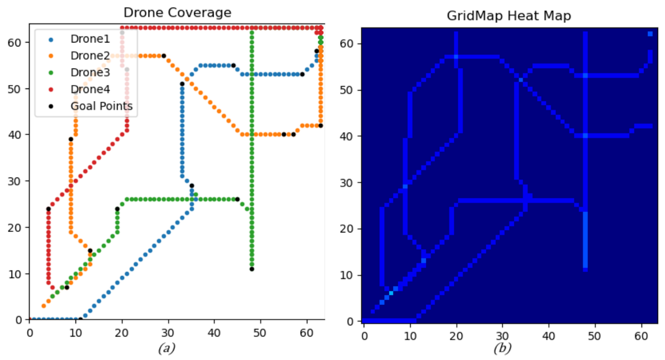


Fig. 7: MC trial with weights $w_a = 0$, $w_g = 1$, $w_e = 0$

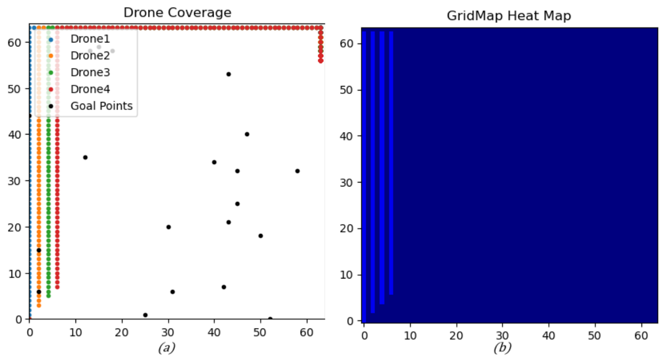


Fig. 8: MC trial with weights $w_a = 0$, $w_g = 0$, $w_e = 1$

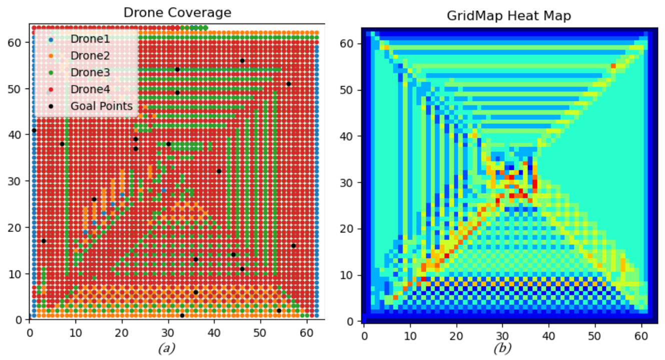


Fig. 9: MC trial with the best overall performance

Here, we extend to a larger area (100x100) with five payload classes and a swarm size of 48 UAVs.

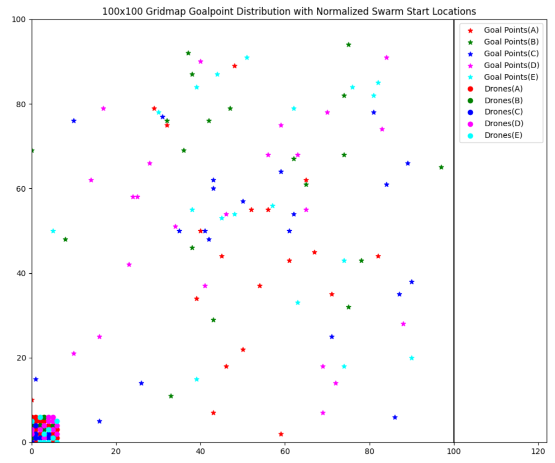


Fig. 10: Normalized swarm initial locations with mission point distribution

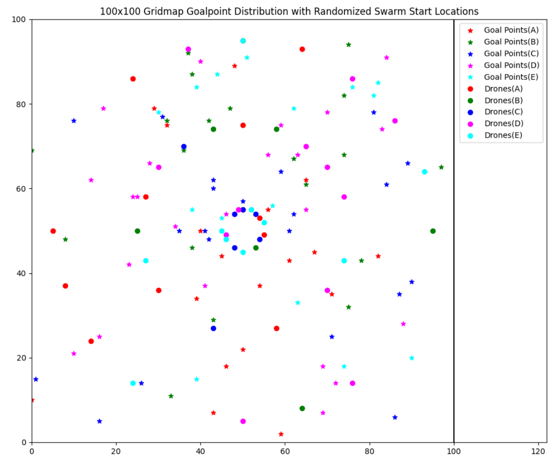


Fig. 11: Randomized swarm initial locations with mission point distribution

There are two starting configurations for the swarm. The first is a pre-deployment starting location: all of the swarm agents are starting off in a cluster at the edge of the map, as if being deployed out into the environment for the first time (i.e. "first flight" configuration). The other configuration has each of the swarm agents starting in randomized locations spread out throughout the environment; this is to simulate a pre-deployed swarm obtaining a new mission after operating for some time, or a pre-distributed swarm receiving its first mission orders.

VI. Discussion

After several iterations of tuning and experimentation with the HD-RHC controller, we were able to discover the impact of each of the weights on the controller, and better categorize how to configure the HD-RHC controller for different mission approaches. The system allows for a variable number of heterogeneous agent types to be considered and can optimize for different mission costs (efficiency, coverage, mission-completion speed). The integration of this controller with AirSim allowed training and evaluation of the system using high-fidelity UAV odometry. This provided a realistic understanding of the dynamics present within these autonomous systems.

A. Mission Completion Analysis

Equation (15) describes a cost function balancing three different trade-offs. Understanding the dynamics between each is crucial to understanding phenomena whilst tuning the controller. For example, in Figure 5, odometry (a) shows mission point visitation, but demonstrates a missed region of coverage; the heat map (b) demonstrates contention in favoring which direction to expand search coverage as highlighted by the hot/red rows & columns. Intuitively, costs focusing exclusively one one of the three demonstrate mission focus in that area exclusively: non-zero weight for w_a maximizes area coverage; non-zero weight for w_g maximizes mission point visitation; non-zero weight for w_e favors momentum preservation.

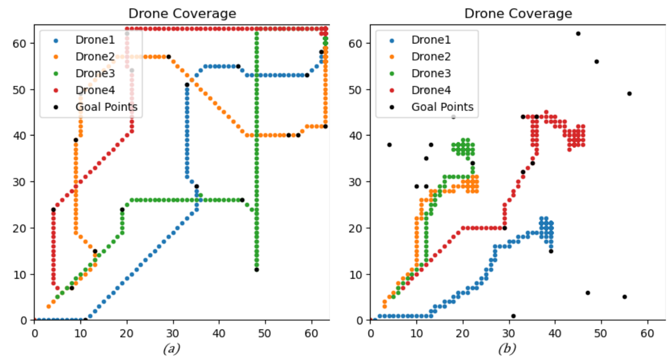


Fig. 12: Odometries caused by slight perturbation of w_a

Here, we noticed certain effects: particularly when $w_a \geq 0.1$, we begin to observe mission points never being explored. Further analysis of the odometry noted that the controller would oscillate between favoring a mission point visitation or favoring search coverage exploration near $w_a \sim 0.1$. An interesting revelation is how sensitive the controller was to changes for each of the weights; changing one weight would cause unexpected interference with the other respective weight parameters. As seen in Figure 12, by adding a minor weight to search area coverage cost, the swarm begins to miss mission points entirely and struggles by alternating its focus between mission point visitation and coverage, essentially leading to a standstill. From this, we conclude that naive uninformed searches cannot produce optimal tuning without exhaustive highly-resolute searches. Better calibrations are possible when the weights are first set to cost-normalized values. As a result, when mission costs are changed, these parameters are not easily transferable. Nonetheless, when a new cost function is considered holistically, a set of initial weights can be derived.

B. Tuning Scalability

Determining optimal weights with a smaller swarm & mission-scope, then applying them to a larger swarm hasn't been thoroughly analyzed. So after deriving an optimal tuning, we applied it to the large swarm configurations mentioned in the Scalability Experiments section. Below are the results of these trials.

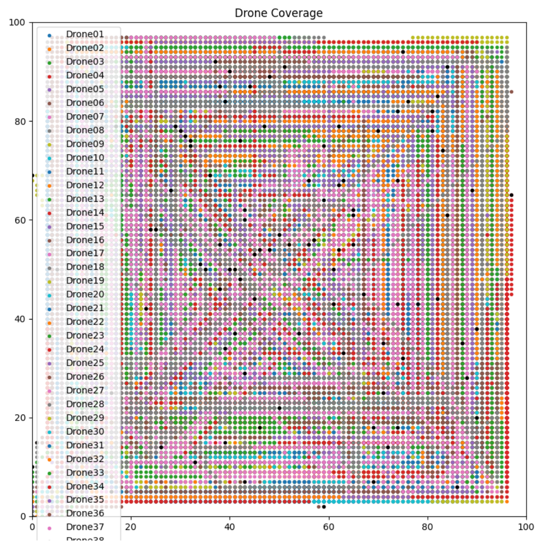


Fig. 13: Coverage with normalized start location

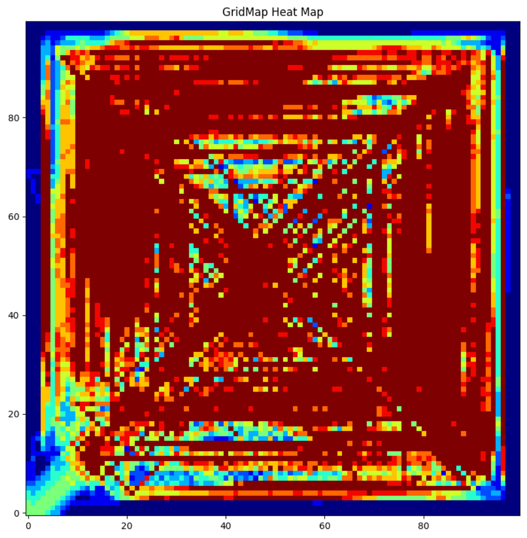


Fig. 14: Swarm efficiency with normalized start location

In both cases, the swarm was able to achieve full search coverage and mission point visitation in roughly equal runtimes (2934 and 3288 time steps, respectively) with the same parameters used in the four UAV / two payload classes normalized configuration. This effectively demonstrates that tuning in simulation need not match the deployed UAV size nor capability. Rather, tuning only needs to focus on striking

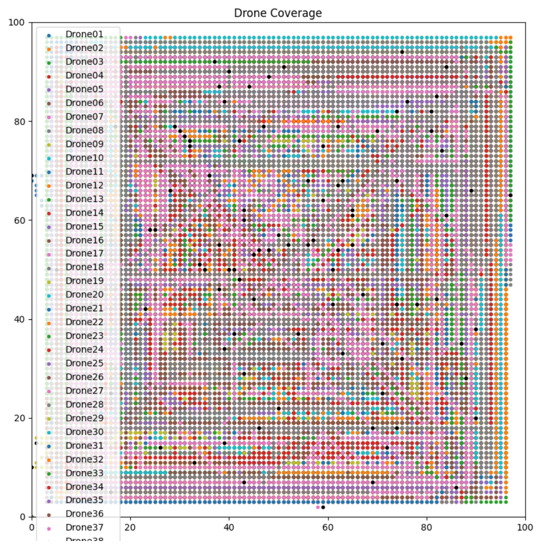


Fig. 15: Coverage with randomized start location

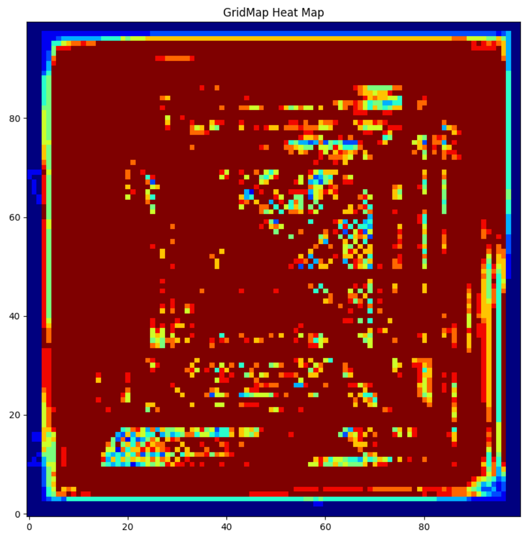


Fig. 16: Swarm efficiency with randomized start location

a balance between objective costs. Therefore, it is possible to increase training efficiency by using only a handful of models instead of a comprehensive modeling of the target swarm.

VII. Conclusion

In this paper, we were able to build on existing work using receding horizon control to manage multiple UAV agents, by introducing a new type of controller: Heterogeneous Decentralized Receding Horizon Control (HD-RHC). With HD-RHC, we are now able to provide scheduling and planning for a heterogeneous UAV swarm in both a known and unknown environment. Each platform can be assigned a unique payload, which correlates to specific mission capabilities (sensing, actuation, etc.). Scheduling and planning work the same as a standard RHC system, but now include a scalable number of unique platforms and mission capabilities. This controller also considers communication constraints, currently designed for use with a mesh network of variable strength. This system is able to be trained both headless, using basic graph representation for the search space and UAV positions, as well as with the AirSim high-fidelity simulation environment. Through initial tuning and testing, a small range of weights has been selected that allow for fast mission complete time and full search coverage of unknown areas. The sensitivity of the controller to the weights allows for a high level of mission adaptability; with proper tuning, the HD-RHC controller can be used for a number of different mission outcomes, such as large-scale area exploration (search and rescue), as well as mission-focused deployments (i.e. package delivery). The controller and training environment will be available for open-source use on GitHub.

VIII. Future Work

Though the controller has demonstrated effectiveness for heterogeneous swarms, we understand that this proof of concept is limited. To explore the true dynamics of this controller, we intend to introduce higher fidelity simulation models. This includes modeling varying payload energy and operational costs: various sensors require different lengths of time to properly measure certain phenomena (e.g., gas leak detection). This in turn would require changes to the controller's cost function to account for non-uniform visitation time costs.

Another avenue of future work is to implement and test the HD-RHC controller with several other simulation platforms. Of interest are the NASA ICAROUS system, due to its support for low-powered systems and formally verified architecture, and the new AWS RoboMaker simulation environment, which would provide extended support for tuning huge-number systems using cloud services [40]. Additional areas we wish to explore are extending beyond the one-hot membership function. There are two additional combinations of payload classes to consider: payloads which have multiple mission applicabilities, and payloads which have partial mission applicability. In the former, we expect that the energy and time costs of motion for such a UAV to be greater than their dedicated-payload counterparts, thus needing to expand mission point selection to consider more than distance, and reliability scores. In the case of partial-membership, we expect the mission selection bids to be factored by applicability, but know that tuning is required to not extensively favor full-membership payloads.

The modeling of the controller itself assumes discrete actions and synchronous control epochs. We wish to further explore applications in continuous action space and non-synchronized planning. Doing so will more closely model real-world implementations, but also may change the cost function model to consider the probability mass of selected actions of neighboring UAVs in the swarm.

Acknowledgment

The authors of this paper would like to thank the faculty and students of the Interactive Controls Laboratory and the NSF SHREC Center at the University of Pittsburgh for their continued support, insights, and resources provided for this research project. We would also like to thank the AirSim team at Microsoft Research, and specifically the Aerial Informatics and Robotics Group for their ongoing support and the development of the AirSim tool.

References

[1] C. Sampedro, H. Bavle, J. L. Sanchez-Lopez, R. A. S. Fernández, A. Rodríguez-Ramos, M. Molina, and P. Campoy, "A flexible and dynamic mission planning architecture for uav swarm coordination," in *2016 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, 2016, pp. 355–363.

[2] J. L. Vian, A. R. Mansouri, and E. W. Saad, "System and method for inspection of structures and objects by swarm of remote unmanned vehicles," Nov. 15 2011, uS Patent 8,060,270.

[3] M. Ma and Y. Yang, "Adaptive triangular deployment algorithm for unattended mobile sensor networks," *IEEE Transactions on Computers*, vol. 56, no. 7, pp. 946–847, 2007.

[4] E. Şahin, "Swarm robotics: From sources of inspiration to domains of application," in *International workshop on swarm robotics*. Springer, 2004, pp. 10–20.

[5] J. C. Barca and Y. A. Sekercioglu, "Swarm robotics reviewed," *Robotica*, vol. 31, no. 3, pp. 345–359, 2013.

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

[7] S. Berman, Q. Lindsey, M. S. Sakar, V. Kumar, and S. C. Pratt, "Experimental study and modeling of group retrieval in ants as an approach to collective transport in swarm robotic systems," *Proceedings of the IEEE*, vol. 99, no. 9, pp. 1470–1481, 2011.

[8] J. Cortes, S. Martinez, T. Karatas, and F. Bullo, "Coverage control for mobile sensing networks," *IEEE Transactions on robotics and Automation*, vol. 20, no. 2, pp. 243–255, 2004.

[9] K. Lerman, A. Martinoli, and A. Galstyan, "A review of probabilistic macroscopic models for swarm robotic systems," in *International workshop on swarm robotics*. Springer, 2004, pp. 143–152.

[10] Y. U. Cao, A. S. Fukunaga, and A. Kahng, "Cooperative mobile robotics: Antecedents and directions," *Autonomous robots*, vol. 4, no. 1, pp. 7–27, 1997.

[11] J. C. Barca, G. Rumantir, and R. Li, "A concept for optimizing behavioural effectiveness & efficiency," in *Intelligent Engineering Systems and Computational Cybernetics*. Springer, 2009, pp. 449–458.

[12] S. Martinez, J. Cortes, and F. Bullo, "Motion coordination with distributed information," *IEEE control systems magazine*, vol. 27, no. 4, pp. 75–88, 2007.

[13] J. Penders, L. Alboul, U. Witkowski, A. Naghsh, J. Saez-Pons, S. Herbrechtsmeier, and M. El-Habbal, "A robot swarm assisting a human fire-fighter," *Advanced Robotics*, vol. 25, no. 1-2, pp. 93–117, 2011.

[14] J. Colorado, A. Barrientos, C. Rossi, and J. del Cerro, "Follow-the-leader formation marching through a scalable o (log 2 n) parallel architecture." in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2010, pp. 5583–5588.

[15] N. Heo and P. K. Varshney, "Energy-efficient deployment of intelligent mobile sensor networks," *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, vol. 35, no. 1, pp. 78–92, 2004.

[16] Y. Tian, K. Liu, K. Ok, L. Tran, D. Allen, N. Roy, and J. P. How, "Search and rescue under the forest canopy using multiple uas," in *International Symposium on Experimental Robotics*. Springer, 2018, pp. 140–152.

[17] J. Cortés and F. Bullo, "Coordination and geometric optimization via distributed dynamical systems," *SIAM journal on control and optimization*, vol. 44, no. 5, pp. 1543–1574, 2005.

[18] A. Howard, M. J. Matarić, and G. S. Sukhatme, "Mobile sensor network deployment using potential fields: A distributed, scalable solution to the area coverage problem," in *Distributed Autonomous Robotic Systems 5*. Springer, 2002, pp. 299–308.

[19] S. Mian, T. Garrett, A. Glandon, C. Manderino, S. Balachandran, C. Dolph, and C. A. Munoz, "Autonomous spacecraft inspection with free-flying drones," in *2020 IEEE/AIAA 37th Digital Avionics Systems Conference (DASC)*. IEEE, 2020.

[20] S. Mian, "A novel battery management & charging solution for autonomous uav systems," Master's thesis, ARIZONA STATE UNIVERSITY, 2018.

[21] J. Mattingley, Y. Wang, and S. Boyd, "Receding horizon control," *IEEE Control Systems Magazine*, vol. 31, no. 3, pp. 52–65, 2011.

[22] T. Keviczky, K. Fregene, F. Borrelli, G. J. Balas, and D. Godbole, "Coordinated autonomous vehicle formations: decentralization, control synthesis and optimization," in *2006 American Control Conference*. IEEE, 2006, pp. 6–pp.

[23] T. Keviczky, F. Borrelli, and G. J. Balas, "Decentralized receding horizon control for large scale dynamically decoupled systems," *Automatica*, vol. 42, no. 12, pp. 2105–2115, 2006.

[24] T. Keviczky, F. Borrelli, K. Fregene, D. Godbole, and G. J. Balas, "Decentralized receding horizon control and coordination of autonomous vehicle formations," *IEEE Transactions on Control Systems Technology*, vol. 16, no. 1, pp. 19–33, 2007.

[25] P. Henderson, M. Vertescher, D. Meger, and M. Coates, "Cost adaptation for robust decentralized swarm behaviour," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 4099–4106.

[26] L. Ingber, "Adaptive simulated annealing (asa): Lessons learned on simulated annealing applied to combinatorial optimization," *Control Cybern. https://doi. org/10.1115/2777*, 1996.

[27] F. Miao, S. Han, S. Lin, J. A. Stankovic, D. Zhang, S. Munir, H. Huang, T. He, and G. J. Pappas, "Taxi dispatch with real-time sensing data in metropolitan areas: A receding horizon control approach," *IEEE Transactions on Automation Science and Engineering*, vol. 13, no. 2, pp. 463–478, 2016.

[28] B. Zhang, X. Sun, S. Liu, and X. Deng, "Adaptive differential evolution-based receding horizon control design for multi-uav formation reconfiguration," *International Journal of Control, Automation and Systems*, vol. 17, no. 12, pp. 3009–3020, 2019.

[29] F. Koohifar, A. Kumbhar, and I. Guvenc, "Receding horizon multi-uav cooperative tracking of moving rf source," *IEEE Communications Letters*, vol. 21, no. 6, pp. 1433–1436, 2016.

[30] Y. Emam, S. Wilson, M. Hakenberg, U. Munz, and M. Egerstedt, "A receding horizon scheduling approach for search & rescue scenarios," *arXiv preprint arXiv:2004.02347*, 2020.

[31] E. Ebeid, M. Skriver, K. H. Terkildsen, K. Jensen, and U. P. Schultz, "A survey of open-source uav flight controllers and flight simulators," *Microprocessors and Microsystems*, vol. 61, pp. 11–20, 2018.

[32] M. Consiglio, C. Munoz, G. Hagen, A. Narkawicz, and S. Balachandran, "Icarous: Integrated configurable algorithms for reliable operations of unmanned systems," in *2016 IEEE/AIAA 35th Digital Avionics Systems Conference (DASC)*. IEEE, 2016, pp. 1–5.

[33] S. Shah, D. Dey, C. Lovett, and A. Kapoor, "Airsim: High-fidelity visual and physical simulation for autonomous vehicles," in *Field and service robotics*. Springer, 2018, pp. 621–635.

[34] C. Y. Ho, S. Y. Tseng, C. F. Lai, M. S. Wang, and C. J. Chen, "A parameter sharing method for reinforcement learning model between airsim and uavs," in *2018 1st International Cognitive Cities Conference (IC3)*. IEEE, 2018, pp. 20–23.

[35] R. Madaan, N. Gyde, S. Vemprala, M. Brown, K. Nagami, T. Taubner, E. Cristofalo, D. Scaramuzza, M. Schwager, and A. Kapoor, "Airsim drone racing lab," *arXiv preprint arXiv:2003.05654*, 2020.

[36] J. Yick, B. Mukherjee, and D. Ghosal, "Wireless sensor network survey," *Computer networks*, vol. 52, no. 12, pp. 2292–2330, 2008.

[37] P. J. Van Laarhoven and E. H. Aarts, "Simulated annealing," in *Simulated annealing: Theory and applications*. Springer, 1987, pp. 7–15.

[38] J. Hurtado and A. Barbat, "Monte carlo techniques in computational stochastic mechanics," *Archives of Computational Methods in Engineering*, vol. 5, no. 1, p. 3, 1998.

[39] P. Brandimarte, *Handbook in Monte Carlo simulation: applications in financial engineering, risk management, and economics*. John Wiley & Sons, 2014.

[40] H. Zhang and L. Zhang, "Cloud robotics architecture: trends and challenges," in *2019 IEEE International Conference on Service-Oriented System Engineering (SOSE)*. IEEE, 2019, pp. 362–3625.