Received November 15, 2018, accepted November 30, 2018, date of publication December 7, 2018, date of current version December 31, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2885455

Hierarchal Application of Receding Horizon Synthesis and Dynamic Allocation for UAVs Fighting Fires

JOSHUA A. SHAFFER, ESTEFANY CARRILLO, AND HUAN XU, (Member, IEEE)
Department of Aerospace Engineering, University of Maryland at College Park, College Park, MD 20742, USA

Corresponding author: Joshua A. Shaffer (jshaffe9@terpmail.umd.edu)

This work was partially funded by Lockheed Martin and NAVAIR.

ABSTRACT This paper explores the design of a high-level mission planner and controller for managing unmanned aerial vehicles (UAVs) fighting a wildfire through the utilization of reactive synthesis and dynamic allocation of the UAVs as resources for the fire. The contribution of this paper is a study on the hierarchal integration of reactive synthesis, used for assuring desired system design traits, and dynamic allocation, used for making heuristic-based decisions. Reactive synthesis provides a formal means of guaranteeing the UAVs' transition to areas of fire, refill of water, and land as defined by the linear temporal logic specifications. Dynamic allocation coordinates the behavior of multiple UAVs through assignments to regions of fire based on a cost function that takes into consideration the fire locations relative to a UAV, distance to the domain edge, wind speed and direction, and the amount of suppressant already present. The use of receding horizons in the reactive synthesis formulation incorporates horizons defined only through spatial distance from a goal. Modifications to these horizon definitions guarantee that the scenario still maintains the overall realizability of the formal specifications after the inclusion of static obstacles. This paper shows the effectiveness of multiple UAV fleets in slowing down the progression of fires from reaching the domain edge through six fire scenarios. At last, our results and successful application demonstrate the utilization of reactive synthesis in larger task spaces and the implications of abstracting UAV transitions for use in formal methods.

INDEX TERMS Reactive synthesis, receding horizon, temporal motion planning, distributed controllers, unmanned aerial vehicles (UAVs), aerial fire suppression.

I. INTRODUCTION

Reactive synthesis, a means for generating controllers of complex systems using formal specifications, has seen a surge in research over the past decade. The primary benefit and motivation for research using this method is the correct-by-construction attribute: the synthesized controllers take into account system and environment variables with varying dynamics and initial conditions, and are guaranteed to meet the designed progress and safety specifications for a system, assuming the environment behaves as formally described. Hence, programmers are not required to "handcraft" individual behaviors of a system under specific conditions (of which are often error prone) and can instead focus on defining the system and specifications in relation to an environment. The major difficulty of using reactive synthesis is the computational burden in dealing with large numbers of environment and system variables, especially observable when considering dynamic environments as discussed in [1].

Consider, for example, a robot planning problem involving a 10×10 grid in which obstacles could appear at any location. A synthesized controller would need to account for 2^{100} permutations, resulting in an inordinate computation time. Unfortunately, real-world problems can easily involve this scenario's scale of permutations in the environment, hence the difficulty of using reactive synthesis. In many cases, this type of problem would not be tackled by using reactive synthesis and instead handled by an algorithm developed for such a task. When using such a tailor-made algorithm, though, the correct-by-construction attribute is lost alongside the benefit of incorporating larger pieces of the system design from the start. Therefore, it is still beneficial to explore methods in which the majority of the design is handled by reactive synthesis and integrated with simpler methods of handling large environment definitions. For this paper, we examine a relevant application, specifically involving fighting wildfires, in which the objective space (i.e. the progress specifications tied to environment moves) is expansive and the total objective does not have strict bounds on what constitutes success versus failure.

Current methods for fighting fires involve numerous ground workers and piloted vehicles, including aerial vehicles capable of dropping large amounts of suppressant over regions of fire. The economic burden of wildfires on the United States (in 2016 $US) exceeds $63.5 billion annually due to damages, and more than $7.6 billion is spent annually to fight said fires [2]. Aerial vehicles with suppression capabilities have served as critical tools in slowing down the growth of fires due to their far greater range of maneuverability when compared to ground crews [3]. Often these aircraft can change the outcome of a wildfire if used to attack a small fire early enough. As discussed in [4] and explored in [5], unmanned aerial vehicles (UAVs) pose a huge benefit to traditional firefighting methods, with the foremost advantage of creating additional ''eyes in the sky'' and supply drops. These use cases have motivated numerous agencies to explore such options in the last decade. In terms of direct suppression, automated UAVs were used to extinguish fires on their own as was mechanically demonstrated on a small, single-UAV scale in [6] and through Lockheed Martin's use of the K-MAX and Stalker XE in [7]. The creation of a fleet of mid-sized autonomous drones for quick response to fledging small-scale wildfires, in turn, could limit the number of required personale to a site and produce more efficient results. Given this problem concept, a formal description of such would require a large number of environmental variables due to the chaotic behavior of the fire. This serves as an apt example of the type of problem many reactive synthesis-based research endeavors tend to avoid.

When the high-level design of a system does involve large scale environmental permutations and state spaces, solutions typically seek to discretize the synthesis problem or approach the problem from a different perspective. Discretization appears in the use of receding horizon control in [8] and the use of decentralized controllers for multiple agents in [9]. Both examples break down the top-level synthesis problem into smaller, discrete pieces for the computation benefits. On the other hand, [10] approached their synthesis problem with a focus on resolving deadlock under specific environment conditions instead of directly avoiding dynamic obstacles. In each of the presented cases, the problem description focused on a limited task space (i.e. the number of progress goals) and how the solution can handle larger sets of actions from an environment in relationship to safety specifications. The maximum state space size of [8]–[10] were up to the order of 100 variables (for [10] specifically), but the task space for each scenario explored never exceeded 4 progress statements.

For the purposes of fighting large, dynamic fires with a single autonomous UAV system represented by a state space abstraction size of 2 orders in magnitude or greater, a high-level controller created with just reactive synthesis could quickly present an impractical solution if the design should accommodate a fairly granular environment space. Further expanding this concept to a whole fleet of UAVs, the problem worsens due to an increase in the number of system variables proportional to or greater than the number of UAVs, if considering centralized controllers. Even with decentralized controllers for each UAV, additional system variables might need to be introduced to describe coordination and behaviors between the decentralized UAV controllers. To alleviate the computational complexity on reactive synthesis in this regard, the coordination of UAVs can be handled by a dynamic allocation process. Dynamic allocation presents an autonomous method for assigning resources in an ever-changing environment. For example, in [11], the resource allocation problem is framed as a multi-objective optimization problem of minimizing the extinguishing time and resource utilization cost, solved by the use of evolutionary algorithms. In [12], the fire behaviors and results due to resource allocation are formulated as a Markov Decision Process, from which a Monte Carlo tree search is used to determine the best areas to allocate resources.

We examine a simplified optimization allocation strategy to address the assignments of UAVs for our fire fighting scenario. If the fleet of UAVs are treated as resources to manage with respect to a changing fire landscape, dynamic allocation would serve well in assigning the UAVs to specific fires. Assignments would depend upon factors in the behavior of the fires such as density, ability to spread, wind and direction, and more. An allocation algorithm would not necessarily be constructed to control the UAVs within the state space defined, nor would the algorithm manage other high-level system aspects associated with the UAVs, such as suppressant control and decisions on landings. Building these high-level behaviors into an allocation algorithm requires ''handcrafting'' these behaviors for all scenarios, an approach that synthesis, on the other hand, is well suited to avoid.

To integrate the two discussed methods-reactive synthesis and dynamic allocation-we utilize a receding horizon framework for reactive synthesis as discussed in [1]. As far as we have found, [13] first touched upon the idea of manipulating the receding horizon framework for decomposing the synthesized problem and decentralizing the planning procedure. For their purposes, this idea resulted in the ability of multiple agents to satisfy high-level specifications through only considering other agents that entered their local horizon. For our purposes, decentralizing the planning procedure allows for dynamic allocation to arrange the order of progress goals specified in the synthesized controller in real-time. Through this, we seek to reconcile the strengths and weaknesses of the two discussed methods to create a high-level mission planner and controller for implementation in a fire fighting scenario.

The contributions of our paper are as follows. First, the implemented receding horizon modification and dynamic allocation demonstrates combining correct-by-construction designs hierarchically with heuristic-based methods. Next, we introduce an algorithm (and proof) for modifying horizons initially defined only by spatial distance to preserve the realizability of the total specification, enabling the application of the template to all goals while including static obstacles. Last, we present results and discussions on the effectiveness of our solution to a simulation environment that uses standard wildfire models. The paper is structured as such. In *Section II*, we present subjects pertinent to the exploration of our topic, primarily in relation to reactive synthesis. In *Section III*, we present the fire fighting scenario, including environment and system definitions. *Section IV* discusses our proposed solution, followed by the implementation of our solution in *Section V*. In *Section VI*, we present the outcomes to our tested cases, followed by our conclusions in *Section VII*.

II. PRELIMINARIES

Linear temporal logic (LTL) is utilized for describing specifications within the reactive synthesis framework. LTL makes use of boolean system variables that serve as atomic propositions (AP), and LTL propositional formulas are built through APs with logic connections and temporal modal operators. Logic connections include $\neg$ (negation), $\vee$ (or), $\wedge$ (and), and $\Longrightarrow$ (implication). Temporal modal operators include $\bigcirc$ (next), $\square$ (always), $\Diamond$ (eventually), and $\mathcal{U}$ (until). Through the use of LTL, a broad range of specifications can be written to describe the behaviors of a system or environment. We point the reader to the preliminaries section of [14] for an expanded description of LTL as it pertains to our purposes.

Reactive synthesis provides a method for generating controllers within the context of a defined environment and system as specified through LTL. The assume-guarantee form of (1) is one of the most commonly used forms for these LTL formulas due to its proven polynomial synthesis time instead of exponential. This equation is also called general reactivity(1) and described further in [14],

$$(\varphi^e_{init} \wedge \bigwedge_{i\in I_r} \square\varphi^e_{s,i} \wedge \bigwedge_{i\in I_f} \square\Diamond\varphi^e_{p,i}) \longrightarrow (\varphi^s_{init} \wedge \bigwedge_{i\in I_s} \square\varphi^s_{s,i} \wedge \bigwedge_{i\in I_g} \square\Diamond\varphi^s_{p,i}). \quad (1)$$

From the above equation, the propositional formula φ_{init} describes the initial condition of the environment or system (denoted by superscript e or s, respectively), $\varphi_{s,i}$ describes safety specifications, and $\varphi_{p,i}$ describes progress specifications.

Reference [1], [15], and [16] have explored reactive synthesis within a receding horizon (RH) framework. The primary benefit of utilizing RH is the segmentation of the state space for both the environment and system into separate horizons. Each horizon provides a smaller problem for which to synthesize a controller, and the combination of these controllers forms a single controller that obeys the specifications written for the total system and environment. The primary disadvantage of RH is that while each horizon itself can be optimized, the total space is not. Other sources have explored methods of optimizing control with respect to time-based rewards on each horizon, such as in [16], but such optimization is beyond the scope of this paper.

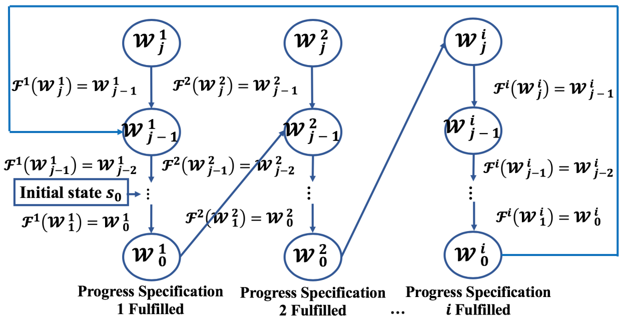

w^1_j
$\mathcal{F}^1(w^1_j) = w^1_{j-1}$
w^1_{j-1}
$\mathcal{F}^1(w^1_{j-1}) = w^1_{j-2}$
Initial state s_0
$\mathcal{F}^1(w^1_1) = w^1_0$
w^1_0
w^2_j
$\mathcal{F}^2(w^2_j) = w^2_{j-1}$
w^2_{j-1}
$\mathcal{F}^2(w^2_{j-1}) = w^2_{j-2}$
$\mathcal{F}^2(w^2_1) = w^2_0$
w^2_0
w^i_j
$\mathcal{F}^i(w^i_j) = w^i_{j-1}$
w^i_{j-1}
$\mathcal{F}^i(w^i_{j-1}) = w^i_{j-2}$
$\mathcal{F}^i(w^i_1) = w^i_0$
w^i_0
Progress Specification 1 Fulfilled
Progress Specification 2 Fulfilled
...
Progress Specification i Fulfilled

FIGURE 1. Segmentation of state space and example of ordered set flow-down performed in receding horizon framework, as described in [1].

For each progress specification, the implementation of RH segments the total system state space into regions $\mathcal{W}_j$ so that, when placed into properly constructed ordered sets $\mathcal{F}^i(\mathcal{W}^i_j)$, the system variables will converge to meeting each progress statement for the system progress goal at $\mathcal{W}_0$. This is shown in Fig. 1. Here, i represents the system progress statement $i \in I_g$, and j indexes the ordered regions $\mathcal{W}$ about the progress specification i. Following the basis laid out by [1], each region consists of its own GR(1) specification (shown in (2)), constructed so that the synthesized controller will move the system states towards the next region within the ordered set (eventually leading to $j = 0$) and fulfill the top level GR(1) specification,

$$\Psi^i_j = ((s \in \mathcal{W}^i_j) \wedge \Phi \wedge \bigwedge_{i\in I_r} \square\varphi^e_{s,i} \wedge \bigwedge_{i\in I_f} \square\Diamond\varphi^e_{p,i}) \longrightarrow (\bigwedge_{i\in I_s} \square\varphi^s_{s,i} \wedge \square\Diamond(s \in \mathcal{F}^i(\mathcal{W}^i_j)) \wedge \square\Phi). \quad (2)$$

In (2), s refers to the system state. The formula Φ consists of all limitations on the states of system, preventing the system from making transitions to or initializing within states that are not allowed. This tautology prevents individual synthesized controllers from creating transitions to states that are infeasible for other horizons.

III. PROBLEM FORMULATION

The high-level problem scenario this paper explores is presented as such. A 450-by-450 meter region of flat grassland, segmented by various large-scale obstacles, is experiencing a wildfire. Fires spread from starting regions under fixed environmental conditions (e.g. wind speed and direction) and with any arbitrary initial conditions. A base of operations exists near the edge of the region and contains a fleet of N UAVs for fighting the fire. Fig. 2 visualizes the abstracted region for this problem.

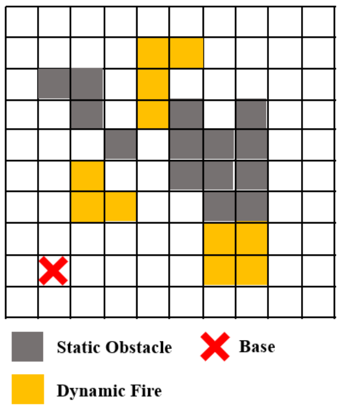


FIGURE 2. 2D grid partition of problem location with environmental indicators.

Each UAV holds a varying level of suppressant for dumping on the fire, from High (100%), Medium (66%), Low (33%), to Empty (0%), associated with a total water volume of $W_v = 125$ liters (requiring a moderately large UAV). Each individual UAV contains a radio for communicating with base, GPS for determining position, and any other sensors required for lower-level controllers. Each UAV's average flight speed, v, is approximately 15 m/s. Design constraints on the UAVs require the need for periodic landing and enforcement of in-flight kinematics resembling fixed-wing behavior. The design goal of this fleet is to significantly slow down the fires' spread to the outer edge of the domain as compared to the fires' natural growth.

The formal definition of the abstracted system space is provided in Definition 1.

Definition 1: The state set is defined as $S = S_p \times S_o \times W$, where the position set is $S_p = \{(1, 1), (1, 2), (2, 1), \ldots (10, 10)\}$, the orientation set is $S_o = \{0°, 90°, 180°, 270°\}$, and the water level set is $W = \{0\%, 33\%, 66\%, 100\%\}$. A single UAV at any given time is represented as an element $s \in S$. For the elements of s, $s_{x,y}$ is used to represent the position tuple (where x and y can take the position values of the tuple), s_o is used to represent the orientation, and w is used to represent the water level.

Note that the above definition implies that each position represents a cell of 45-by-45 meters. Furthermore, viable transitions for the UAVs between elements in the state space are defined assuming 3 transition scenarios, a sped up counterclockwise turn, a sped up clockwise turn, and a straight drive ahead. These transition scenarios result in the following transition system described in Definition 2, visualized in Fig. 3.

Definition 2: The transition relation for the state set S is defined as $T = \{s \rightarrow s' \in R \subseteq S \times S\}$, where elements $s \rightarrow s' \in R$ are defined for each allowable s' per $s \in S$ under the following conditions.

If $s_o = 0°$:

$$s'_{x,y} = \begin{cases} (s_x, s_y), & s'_o = 0° \\ (s_x + 1, s_y), & s'_o = 0° \\ (s_x + 1, s_y + 1), & s'_o = 90° \\ (s_x + 1, s_y - 1), & s'_o = 270° \end{cases}$$

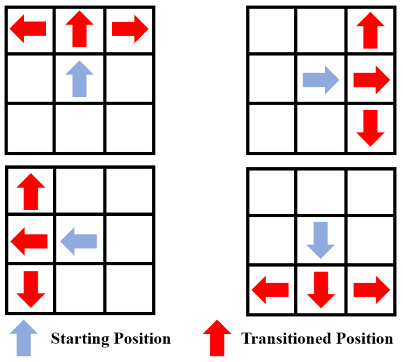


FIGURE 3. Possible transitions for UAV within grid given starting orientation and location.

If $s_o = 90°$:

$$s'_{x,y} = \begin{cases} (s_x, s_y), & s'_o = 90° \\ (s_x, s_y + 1), & s'_o = 90° \\ (s_x + 1, s_y + 1), & s'_o = 0° \\ (s_x - 1, s_y + 1), & s'_o = 180° \end{cases}$$

If $s_o = 180°$:

$$s'_{x,y} = \begin{cases} (s_x, s_y), & s'_o = 180° \\ (s_x - 1, s_y), & s'_o = 180° \\ (s_x - 1, s_y + 1), & s'_o = 90° \\ (s_x - 1, s_y - 1), & s'_o = 270° \end{cases}$$

If $s_o = 270°$:

$$s'_{x,y} = \begin{cases} (s_x, s_y), & s'_o = 270° \\ (s_x, s_y - 1), & s'_o = 270° \\ (s_x - 1, s_y - 1), & s'_o = 180° \\ (s_x + 1, s_y - 1), & s'_o = 0° \end{cases}$$

Depending on the location of static obstacles and boundaries, elements within the described transition relation are restricted from the general case if they violate the *reachability* property of the system, as defined through Definition 3. This implies that the existence of obstacles requires limitation on allowable states S in the system specifications. A graph search for paths that lead to dead ends is an accessible way of determining these states.

Definition 3: The system S is defined as reachable if there exists a path of states, composed of a finite number of subsequent transitions in $s \rightarrow s' \in T \subseteq S \times S$, such that all $s \in S$ can be eventually reached from any other initial state $s \in S$.

The formal definition of the abstracted environment space is provided in Definition 4. Note that this general environment definition is expansive in size (up to 2^{100+N} combinations, where N is the number of fires). Also note that this environment definition serves as an abstraction to a more complicated fire growth model, described in further detail in *Section V*.

Definition 4: The fire environment is defined as $F_{x,y} = (x, y) \times \{True, False\}$ for any valid choice of x and y in the system domain, excluding the base and obstacle locations. The element $f_{x,y} \in F_{x,y}$ corresponds to the presence of fire *(i.e. True or False) associated with the tuple* (x, y). *The total possible environment is the combination of all* $F_{x,y}$ *sets with the landing signal sets of each UAV, i.e.* $E = F_{1,1} \times F_{1,2} \times F_{2,1}, \times \ldots F_{10,10} \times S_{land,1} \times S_{land,2} \times \ldots S_{land,N}$, *where* $S_{land,n} = \{True, False\}$ *corresponds to the landing signal of the* n^{th} *UAV.*

Under these formal definitions on the system and environment, the desired design for each UAV are as follows. First, each UAV must fly to any region in the state space associated with an active fire (Eq. (3)), dumping a fraction of its water supply if possible (Eq. (4)),

$$\varphi_1^s = \bigwedge_{(x',y')} \Box(f_{x',y'} \longrightarrow \Diamond(s_{x,y} \leftrightarrow (x',y'))), \qquad (3)$$

$$\varphi_2^s = \bigwedge_{(x',y')} \Box((w > 0\% \wedge f_{x',y'} \wedge (s_{x,y} \leftrightarrow (x',y'))) \leftrightarrow \bigcirc w = w - 33\%). \qquad (4)$$

Next, each UAV must return to base for replenishing water supplies when empty (Eq. (5)) and the water level must refill to the max level when the UAV reaches base (Eq. (6)). Outside of any condition that forces the water level to change, the water level must remain constant (Eq. (7)). Note that *base* is an AP that is *True* when s_p matches the associated (x, y) tuple for the base,

$$\varphi_3^s = \Box(w = 0\% \longrightarrow \Diamond base), \qquad (5)$$

$$\varphi_4^s = \Box((w = 0\% \wedge base) \leftrightarrow \bigcirc w = 100\%), \qquad (6)$$

$$\varphi_5^s = \Box((! \bigcirc w = 100\% \wedge ! \bigcirc w{=}w{-}33\%) \leftrightarrow \bigcirc w = w). \qquad (7)$$

The UAVs may experience landing signals and must land for prolonged periods of time in the next available region not consumed by fire (8),

$$\varphi_6^s = \Box((s_{land,n} \wedge \neg f_{x,y}) \leftrightarrow (\bigcirc s_{x,y} \leftrightarrow s_{x,y})). \qquad (8)$$

Lastly, the order in which fires are addressed must be prioritized by their capability of reaching the domain edge, and UAVs must allocate themselves in a manner that increases their combined effect on the environment, a specification represented by $\varphi^s_{priority}$. This specification is an open area of design, one in which a formal description of the fire behavior φ^e_{model} and system response $\varphi^s_{priority}$ would require further environment and system definitions, specifically in relation to a model of wildfire dynamics. In the reactive synthesis language, the problem we are attempting to solve is the total specification:

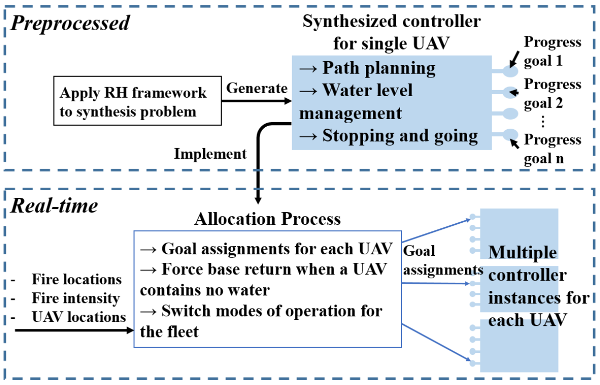
$$\varphi^e_{model} \longrightarrow \bigwedge_{i \in [1,6]} \varphi^s_i \wedge \varphi^s_{priority}. \qquad (9)$$

[Note that the individual specifications in (9), specifically (3), (4), and (5), are not in GR(1) form.]

Given (9) and the desired design, the problem we solve is how to address the specifications φ^e_{model} and $\varphi^s_{priority}$ and "synthesize" a controller for (9). Specifically, how can we avoid introducing further variables to the synthesis problem while capturing the intended behavior of (9) through utilizing the formal tools discussed in the *Preliminaries* section alongside a heuristic-based method of determining which fires the UAVs should prioritize in order to suppress such?

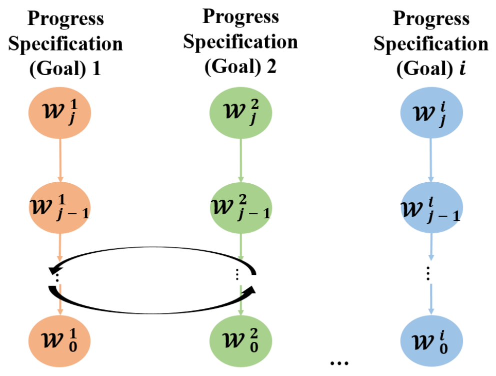
IV. PROPOSED SOLUTION METHOD

We propose a solution method to the formulated problem scenario that combines reactive synthesis with a dynamic allocation algorithm. These two methods form a high-level planner and controller that fulfills the design constraints imposed on each UAV and dictates the behavior of each one as well as their collective maneuvers. Fig. 4 depicts a conceptual view of the process of creating our solution and the duties that each method performs. Fig. 5 depicts the direct relationship between the allocation process and a synthesized controller.



Preprocessed
Synthesized controller for single UAV
→ Path planning
→ Water level management
→ Stopping and going
Progress goal 1
Progress goal 2
Progress goal n
Apply RH framework to synthesis problem
Generate
Implement
Real-time
Allocation Process
→ Goal assignments for each UAV
→ Force base return when a UAV contains no water
→ Switch modes of operation for the fleet
Goal assignments
Multiple controller instances for each UAV
- Fire locations
- Fire intensity
- UAV locations

FIGURE 4. Diagram of creating the solution method and the responsibilities and roles for both the synthesized controllers and the allocation process during real-time implementation.



Progress Specification (Goal) 1
Progress Specification (Goal) 2
Progress Specification (Goal) i
w_j^1
w_j^2
w_j^i
w_{j-1}^1
w_{j-1}^2
w_{j-1}^i
w_0^1
w_0^2
w_0^i
...

FIGURE 5. Diagram of allocation process rearranging the progress goal ordering (as depicted in Fig. 1) for a single UAV controller instance in real-time.

As shown in Fig. 4, a common synthesized controller is created for each of the UAVs through the RH framework with the duties presented. Given any arbitrary initial condition, the controller aims to progress to each viable partitioned space, these progress goals represented within Fig. 4 by the "nodes" protruding from each rectangle. The order that the controller meets these progress statements for a single UAV, shown previously through the ordering of $\mathcal{W}^i$ in Fig. 1, is not dictated by the synthesized controller as typically performed within the RH framework. Instead, the allocation process decides which progress specification any single UAV should pursue, represented within Fig. 5 by the switching of the ''nodes''' order for any given moment in time, and the allocation process is responsible for ensuring that each ''node'' can and/or will be fulfilled. Hence, the allocation process prioritizes and assigns which goals a single controller should meet next in real-time. The combination of these two methods in the described manner works to highlight the strengths of each method. The synthesized controllers manage various system oriented aspects of the design and path planning while allocation governs the fleet behavior through assignments of goals for each controller.

Previously mentioned in [6], a physical scenario was constructed that dealt solely with one UAV gathering water, moving to another location, and dumping said water on the destination. Reference [6] demonstrates the existence of lower level controllers that could manage the individual actions necessary to achieve the high-level planner and controller this paper proposes (at least for a smaller scale UAV). So, as often expressed in other sources dealing with high-level synthesized controllers, we assume there exists low-level controllers to dictate the motion of individual agents in real-time.

V. IMPLEMENTATION

This section describes the construction and operation of the synthesized controllers and allocation algorithm used for the high-level controller. Additionally, the simulation used to test such cases is also outlined.

A. SYNTHESIS OF CONTROLLERS IN RECEDING HORIZON FRAMEWORK

The RH framework discussed generates individual synthesis problems about each progress goal while maintaining all safety specifications as formulated for the entire system. For (3) and (5), translated GR(1) goals involve always eventually driving the system to the regions holding fire or the base, invoked by the presence of fire or absence of held water, respectively. For creation of the actual specifications used in synthesis, we assume that the dynamic allocation process correctly interprets the required conditions which force the UAV to the base or fire (e.g. when the water level is empty, instead of the synthesized controller interpreting such and driving the UAV to base, the dynamic allocation recognizes and forces the UAV to prioritize the goal associated with base). As a result, (3) and (5) are simply reinterpreted as (10) and (11), respectively. Therefore, the specifications used in synthesis will include the safety specifications shown in (4), (6), and (8) alongside these simplified goal specifications,

$$\bigwedge_{(x',y')} \Box\Diamond(s_{x,y} \leftrightarrow (x',y')), \tag{10}$$

$$\Box\Diamond base. \tag{11}$$

For the common synthesized controller, the formal environmental description described beforehand is broken down to the relative signals as perceived by a single UAV and manipulated by either the allocation method or the naturally occurring environment. These Boolean variables consist of: s_{land}, representing whether or not the UAV needs to stop due to a landing signal; f_d, representing the presence of fire directly beneath the UAV; and *drop*, representing the allocation process's signal to a UAV for dropping water on the assigned goal location. Hence, the relative environment $E_r = \{s_{land}, f_d, drop\}$.

The environment specifications for a single UAV are listed as follows:

$$\varphi^e_{init} = \{\}, \tag{12}$$

$$\varphi^e_s = \{\}, \tag{13}$$

$$\varphi^e_p = \Box\Diamond drop \wedge \Box\Diamond\neg s_{land} \wedge \Box\Diamond\neg f_d. \tag{14}$$

Equation (12) and (13) show that no guarantees are provided on the environment's initial condition and safety behavior, and (14) states that always eventually allocation instructs the UAV to drop water, the UAV will experience engine failure, and the UAV will not fly over fire.

The system consists of S as described in Definition 1. Additional APs *goal* and *base* were created for when the UAV enters a specified goal location and the base location, respectively. These revised definitions are used to form updated specifications in GR(1) form for each set of horizons about each goal.

The system specifications from *Section III* were formally defined about each goal tied to a tuple (x, y) for each UAV :

$$\varphi^s_{init} = \Phi. \tag{15}$$

Equation (15) reflects that a UAV will start in a state allowed by Φ (the RH tautology that governs feasible states). In this scenario, the tautology simply prevents the system from occupying any state that violates the conditions of Definition 3 before the horizons have been applied,

$$\varphi^s_{s,1} = \Box((s_{land} \wedge \neg f_d) \leftrightarrow (\bigcirc s_{x,y} \leftrightarrow s_{x,y})). \tag{16}$$

Equation (16) is the modified form of (8), already in GR(1) form, replacing the global $f_{x,y}$ element with the local f_d variable. Equations (17), (18), and (19) are the modified versions of (6), (4), and (7), respectively.

$$\varphi^s_{s,2} = \Box((w = 0\% \wedge goal \wedge base) \leftrightarrow \bigcirc w = 100\%), \tag{17}$$

$$\varphi^s_{s,3} = \Box((w > 0\% \wedge drop \wedge goal \wedge \neg base) \leftrightarrow \bigcirc w = w - 33\%), \tag{18}$$

$$\varphi^s_{s,4} = \Box((!\bigcirc w = 100\% \wedge !\bigcirc w = w - 33\%) \leftrightarrow \bigcirc w = w). \tag{19}$$

In (17), the system reaching the base location has been replaced with the local *goal* $\wedge$ *base* proposition combination to enforce the notion that UAVs only fill up when reaching base if the base was also set as the goal,

$$\varphi_p^s = \Box\Diamond goal. \tag{20}$$

Lastly, (20) is the generalized version of (10) and (11). Equation (20) models both since the allocation process is responsible for ensuring the conditions that drive the UAV to either the base location or a particular fire location, which serve as the current goal.

Equations (15) - (20) along with the environment definitions form (21), the synthesis problem about each progress goal. This formulation in the RH framework captures the intended behavior of the original specifications while enabling the allocation method to dictate which goals are prioritized in which order,

$$\begin{aligned}\Psi^{x',y'} &= \varphi_{init}^e \wedge \varphi_s^e \wedge \varphi_p^e \\ &\longrightarrow \varphi_{init}^s \wedge \varphi_{s,1}^s \wedge \varphi_{s,2}^s \wedge \varphi_{s,3}^s \wedge \varphi_{s,4}^s \wedge \varphi_p^s \wedge \Box\Phi. \end{aligned}\tag{21}$$

To apply the RH framework, the state space must be segmented into horizons $\mathcal{W}_j^{x',y'}$ about every possible goal region. A formal definition for such is provided by Definition 5, and Fig. 6 visualizes such for a single goal.

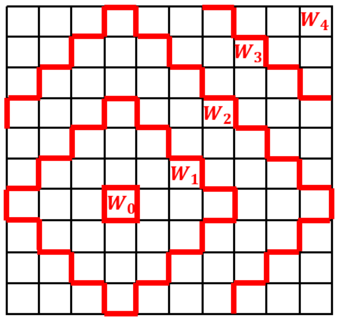


FIGURE 6. Example of RH partitions for progress statement centered on position (4,4), i.e. $\mathcal{W}_0$.

Definition 5: Horizons are defined as $\mathcal{W}_j^{x',y'} = \{s \in S \mid 3(j-1) \leq |s_x - x'| + |s_y - y'| \leq 3j$ *where* $j = 1, 2, 3, \ldots\}$. $\mathcal{W}_0^{x',y'}$ *refers to goal.*

These horizons are utilized to form the individual $\Psi_j^{x',y'}$ specifications used in the RH framework. This formal definition is intuitive and easy to apply to this problem since it requires a simple calculation while automating the synthesis process. Unfortunately, these horizons do not account for static obstacle placements which create the restrictions represented by Φ, and blind application will create unrealizable specifications. A possible case of this issue is shown in Fig. 7, in which transitions from state (1) in $\mathcal{W}_2$ cannot go into $\mathcal{W}_1$ because of obstacles. Transitions that are allowed to "move back" to $\mathcal{W}_3$, however, can subsequently provide a path back to $\mathcal{W}_1$, as represented by the numbered arrows. To circumvent a horizon violation as such, a horizon modification algorithm is applied during synthesis to maintain the realizability of all RH specifications, shown in Algorithm 1.

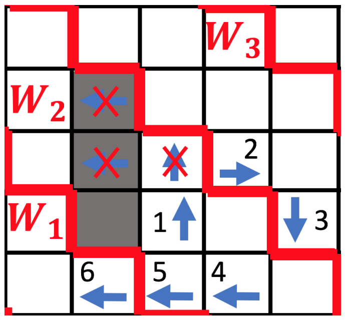


FIGURE 7. Example of an unrealizable state (1) in $\mathcal{W}_2$ for the horizon specification Ψ_2 that does have a valid next move (2) into $\mathcal{W}_3$ and subsequent path back to $\mathcal{W}_1$.

Algorithm 1 $\mathcal{W}_j^{x',y'}$ Modification During Synthesis

```
1: procedure Synthesis_Goal($x', y'$)    ▷ Synthesis controllers from all initial conditions for the $x', y'$ goal location
2:     for $0 \leq j \leq N$ do
3:         for $s \in \mathcal{W}_j^{x',y'}$ do
4:             Synthesize controller given $x', y'$ goal and current $s$
5:             if Controller == None then
6:                 Remove $s$ from $\mathcal{W}_j^{x',y'}$
7:                 Add $s$ to $\mathcal{W}_{j+1}^{x',y'}$
```

1) RECEDING HORIZON MODIFICATION AND PROOF OF SPECIFICATION VALIDITY

Theorem 1: Given a modified version of the system in Definition 1 which has restricted accessible states and transitions through the addition of static obstacles but still maintains the reachability property described in Definition 3, and by using the initial horizons $\mathcal{W}_j^{x',y'}$ *described in Definition 5 applied with the horizon modification algorithm described by Algorithm 1, the specification for each horizon surrounding each progress goal,* $\Psi_j^{x',y'}$*, will remain realizable, preserving the RH framework guarantees on the overall specification.*

Proof: First, we maintain that the overall specification (Eq. (21)) is realizable for the modified system description given no horizons and any allowable initial condition. Because of such, a horizon-based synthesized solution exists that fulfills the framework and definitions provided in [1].

Given the Definition 5 description of the receding horizons $\mathcal{W}_j^{x',y'}$ for any individual goal (x', y'), reachability (Definition 3) implies that for any state sequence π that starts and leads from $s \in \mathcal{W}_j^{x',y'}$ to a state s_f with $s_{f,x,y} = (x', y')$, said sequence π must contain at least one $s \in \mathcal{W}_k^{x',y'}$ for all $0 \leq k \leq j$. Under the modified system definition, all available π sequences for some $s \in \mathcal{W}_j^{x',y'}$ may also need to include $s \in \mathcal{W}_r^{x',y'}$ for some $r > j$, i.e. the only available path to the goal may require the state to move into horizons away from the goal before moving back through horizons towards the goal due to the presence of obstacles. The presence of such a sequence that includes paths with $s \in \mathcal{W}_{r>j}^{x',y'}$ as the only valid path immediately violates the order conditions for the receding horizon specification $\Psi_j^{x',y'}$. To address this violation, modifications to the horizons, as shown in Algorithm 1, are made during synthesis to maintain the condition that a path π does not contain any $s \in \mathcal{W}_{r>j}^{x',y'}$.

As controllers are synthesized around each goal and for each initial condition s_i within each set $\mathcal{W}_j^{x',y'}$, starting with $j = 0$ and incrementing, realizability failures are direct results of the lack of a system path to the horizon $\mathcal{W}_{j-1}^{x',y'}$ that remains only in $\mathcal{W}_j^{x',y'}$. This is a result of the failure to satisfy $\Psi_j^{x',y'}$ since the overall specification $\Psi^{x',y'}$ is realizable. Because a path starting at the initial condition s_i that fulfills the global specification must exist on the global scale and none of the sets $\mathcal{W}_j^{x',y'}$ overlap per index (x', y'), the path must enter into $\mathcal{W}_{j+1}^{x',y'}$ due to the reachability property stated before. Through the algorithm, this state s_i is removed from $\mathcal{W}_j^{x',y'}$ and added to $\mathcal{W}_{j+1}^{x',y'}$. All intermediate states between the initial condition and horizon $\mathcal{W}_{j+1}^{x',y'}$ are also moved to the next horizon since each state is tested as an initial condition in Algorithm 1, and these states cannot serve as viable initial conditions themselves. Therefore, the revised $\mathcal{W}_{j+1}^{x',y'}$ contains the original set $\mathcal{W}_{j+1}^{x',y'}$ plus all states from $\mathcal{W}_j^{x',y'}$ that could not serve as initial conditions to reach the next horizon of $\mathcal{W}_{j-1}^{x',y'}$ (or goal if $j - 1 = 0$). This statement serves as a recursive assignment for each horizon j, shifting states back horizons until a new horizon set for a goal is defined such that each $s \in \mathcal{W}_j^{x',y'}$ starts a path π contained solely in $\mathcal{W}_j^{x',y'}$ that reaches $\mathcal{W}_{j-1}^{x',y'}$. Because of this, $\Psi_j^{x',y'}$ is realizable for all goals, all horizons, and all initial conditions, maintaining the guarantees provided by the RH framework used from [1]. □

A benefit of the approach used by Algorithm 1 is that the viability test for an initial condition is made during synthesis and construction of controllers. If no controller from the initial condition is found that satisfies the specification, the approach automatically generates the needed modifications to the horizon template to render the specification realizable. Doing so during execution allows for keeping valid horizons initially provided and only changing them as needed.

B. DYNAMIC ALLOCATION

In this paper, we propose a straightforward method for managing the dynamic allocation of UAVs to fire perimeter locations that spread with time. The allocation process considers four main attributes that directly affect a fire's progress to the boundary edge and a single UAV's ability to suppress such. These include: proximity of the UAVs to fire locations; proximity of fires to the domain boundary; wind direction and magnitude; and the amount of burn-through time provided by any suppressant acting on the fire. These attributes were chosen due to their simplicity in calculation and their ease of observation outside of the simulation environment. We assume that the fire model (which determines the previous attributes and is explained further in the *Simulation* subsection) and its corresponding perimeter is correctly abstracted into the cells used by the synthesis process (i.e. any cell that contains the fire perimeter is interpreted as holding fire and is made available for allocation).

Algorithm 2 UAV dynamic allocation process

```
1:  procedure Assign_UAVs($UAVs$, $F_p$)    ▷ Allocate all UAVs to fires in the fire perimeter set $F_p$
2:      for $s_{i,(x,y)} \in UAVs$ do
3:          $min\_cost \leftarrow$ inf
4:          for $f_p \in F_p$ do
5:              $cost \leftarrow g(f_p, s_{i,(x,y)})$
6:              if $cost \leq min\_cost$ then
7:                  $f\_min \leftarrow f_p$
8:                  $min\_cost \leftarrow cost$
9:          Allocate $i^{th}$ UAV to $f\_min$
10:         Remove $f\_min$ from $F_p$
```

Algorithm 2 shows the process for allocating UAVs to the fire perimeter locations, performed at each update of the synthesized controllers. The inputs consist of *UAVs*, the set of all location tuples $s_{i,(x,y)}$ for the UAVs (ordered by the index i), and F_p, the set of all current fire perimeter location tuples f_p in the environment domain. Due to the limited number of abstractions that our synthesized controller acts over, calculating the cost of each individual fire relative to each UAV is a feasible option of determining the minimized allocation cost per UAV at each update of the synthesized controllers. Note that no optimization of UAV assignments is performed across members, i.e. the UAVs are assigned to fire locations in order and the next UAV cannot be assigned to a fire previously chosen in the main loop. [The removal of fire elements from F_p in the algorithm do not impact the contents F_p in the next algorithm call, however.]

The cost function $g(f, x)$ is calculated as the weighted sum of: the distance between the fire and the UAV; the fire's distance from the closest boundary edge; the alignment of the fire's direction alongside the wind and its magnitude; and the amount of suppressant acting on the fire. This equation is displayed in (22),

$$\begin{aligned} g(f, x) = d_f * \|x - f\| + e_f * \min(\|edge - f\|) \\ -w_f * \|f \cdot wind\| + b_f * suppressant\_time\_left(f), \end{aligned} \tag{22}$$

where d_f, e_f, w_f, and b_f are all heuristic weights for each relevant attribute. The coefficient d_f behaves as a penalty weight for the distance between a UAV location x and fire location f. The coefficient e_f represents a penalty weight on fires that are further from any location along the outer edge of the domain. The coefficient w_f is an importance weight on fires further along the direction of the wind. A greater value contributes negatively to the total cost. Finally, coefficient b_f corresponds to a penalty on the remaining time for suppressant already present at the fire. This increases a relative fire's cost proportionally to the remaining burn-through time. Within all four terms, the variable *edge* represents any location along the domain boundary, the vector *wind* corresponds to a scenario's x and y components of the wind vector, and the function *suppressant_time_left* calculates the remaining burn through time (in seconds) of suppressant at a current fire location, returning zero if no suppressant is present.

These weights represent "knobs" for heuristically tuning the allocation of UAVs to individual fires. Ideally, we aim to have the UAVs prioritize fire perimeters that are moving further along the wind direction and approaching the edges of the domain foremost. The distance from fire and burn-through time terms act to "spread out" the allocation of UAVs when the wind and edge terms are less severe. Through initial testing, we achieved the desired behaviors using standard units by choosing the coefficients d_f, e_f, w_f, and b_f as 0.1, 1.0, 0.1, and 0.02, respectively. For perspective, the four attribute terms in the cost function without coefficient multiplication, using standard units when evaluated, were typically on the order of 100, 100, 10,000, and 1,000, respectively.

C. SIMULATION

To implement the solution and test its ability to meet the problem scenario, the synthesized controllers and allocation algorithm were constructed in Python alongside a simplified fire simulation following the wavelet differential equations and the Rothermel spread equation provided in FARSITE [17]. FARSITE models the fire fronts as propagating wavelets, with the fire spread rate calculated by the Rothermel spread equation serving as the main indication of intensity. Reference [18] provides the valuation of the Rothermel spread equation for various fuel types and climates, shown in Fig. 8. For the simulation, we assumed the fuel type was SH7, which is shrubbery in a dry climate, and approximated the spread rates for 3 levels of wind speed. Table 1 provides these approximations for each wind speed. At each update time for the fire, the calculated spread rate of a fire vertex along the perimeter was also adjusted with a +/−10% standard deviation to provide further variation in the fire front growth across each simulation. Additionally, the fire front model's perimeter was abstracted into distinct fire regions for use by the dynamic allocation process.

TABLE 1. Chosen wind speeds and resulting spread rates of fires for use in simulation scenarios.

Wind Speed (m/s)	Rate of Spread (m/min)
1.0 (Low)	3.0
4.0 (Medium)	18.0
8.0 (High)	48.0

Fire suppression mechanics are a relatively unknown area of research, and simulations typically assume that obstacles

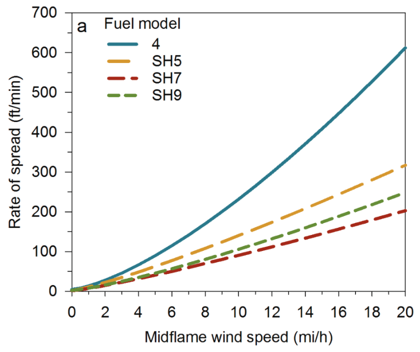


FIGURE 8. Fire spread rates as a function of wind speed for various fuel models, pulled directly from [18].

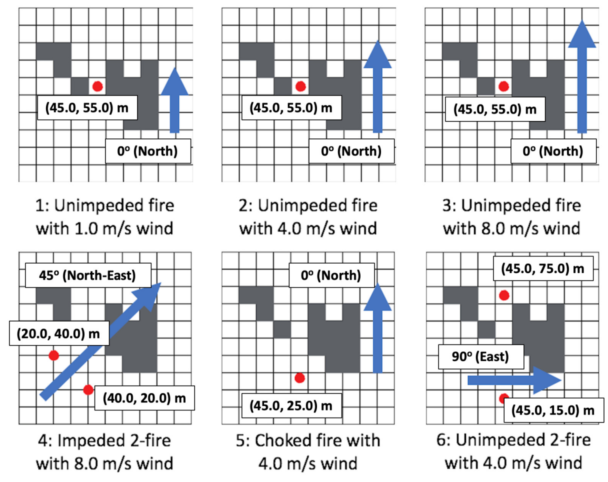

(45.0, 55.0) m
0° (North)
1: Unimpeded fire with 1.0 m/s wind
(45.0, 55.0) m
0° (North)
2: Unimpeded fire with 4.0 m/s wind
(45.0, 55.0) m
0° (North)
3: Unimpeded fire with 8.0 m/s wind
45° (North-East)
(20.0, 40.0) m
(40.0, 20.0) m
4: Impeded 2-fire with 8.0 m/s wind
0° (North)
(45.0, 25.0) m
5: Choked fire with 4.0 m/s wind
(45.0, 75.0) m
90° (East)
(45.0, 15.0) m
6: Unimpeded 2-fire with 4.0 m/s wind

FIGURE 9. Initial conditions of all fire scenarios, showing locations of 25 m diameter starting fires (red dots) and constant wind conditions (blue arrows). Grey boxes represent static obstacles.

(both static and dropped suppressant) simply stop or greatly slow the spread rate at that specific location on a fire front, either indefinitely for static obstacles or a limited time for suppressant (referred to as the burn-through time). The placement of temporary obstacles in our simulation provided the direct mechanisms in which the fire was slowed down at any point by a UAV. [Note that we only modeled slowing the fire down or stopping, no permanent extinguishing of cells.] The dropping of suppressant was modeled after the line length and burn-through times for 1500 liters of suppressant, as discussed in [19]. For a 1500 liter suppressant drop across 45 meters, the burn-through time is approximately 2 hours. We assumed that a linear relationship exists between the amount of suppressant and the burn-through time, i.e. 125 liters constitute about 10 minutes of burn-through, and each fractional drop of 125 liters on an area added the same proportional amount of 10 minutes to the current burn-through time. Additionally, we assumed a linear relationship between the rate of growth of a fire vertex within a suppressant area and the burn-through time left. The endpoints on this linear relationship were 0% of normal growth

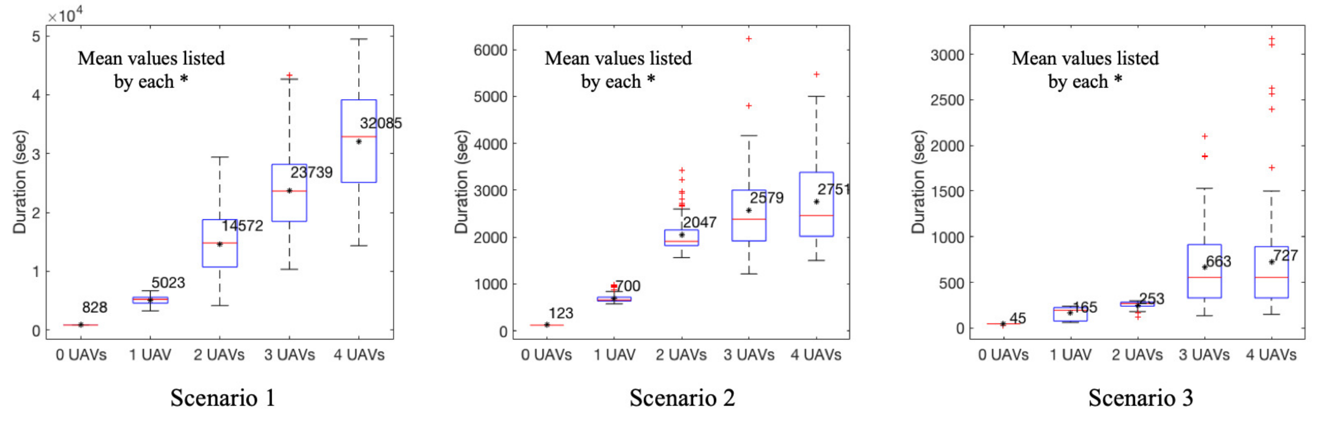


FIGURE 10. **Box plot distributions of the simulated time for fire to reach any domain edge given a set number of UAVs fighting such fires (scenarios 1, 2, and 3).**

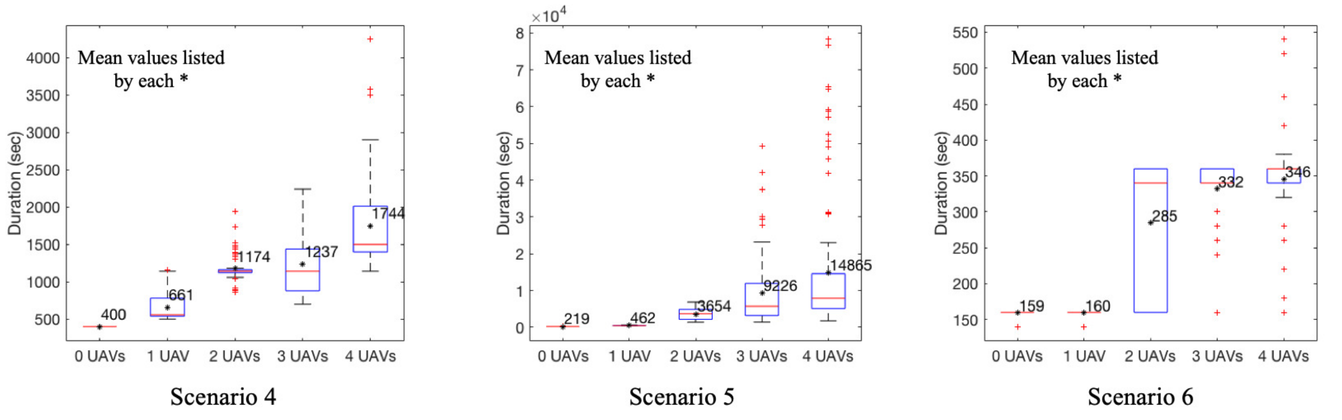


FIGURE 11. **Box plot distributions of the simulated time for fire to reach any domain edge given a set number of UAVs fighting such fires (scenarios 4, 5, and 6).**

rate for full burn-through time left and 5% of normal growth for no burn-through time left. This relationship was included to add conservative stress on the UAVs' ability to suppress the fires. Lastly, the geometry associated with suppressant drops was ignored, i.e. any suppressant dropped on the fire front was assumed to align itself so as to correctly block the fire within that region.

The UAVs were modeled to follow simplified kinematics and assumed to still maintain the transitions described in the problem description at each time step. Additionally, the UAVs were assumed to require 4 minutes anytime they were forced to stop, either by a random stop signal or stopping at the base to pick up suppressant.

Various fire scenarios' initial conditions and parameter settings were constructed for the purposes of testing the controllers. These fire scenarios were aimed at testing the capabilities of the UAVs to slow down all fires from reaching the borders and gauge the effectiveness of different fleet numbers. These scenarios are provided in Fig. 9. The simulation cycled through multiple iterations on each scenario, testing the effectiveness of up to 4 fleet members. The average time for the fire to reach the outer edge on each scenario and fleet number combination was calculated. For all scenarios, the UAVs started at the base location.

VI. RESULTS

TuLiP [20] was utilized to realize and synthesize the controllers associated with each region $\mathcal{W}_j^{x',y'}$. On an Intel i5-6500 CPU @ 3.20 GHz processor, this total process, approximately 250 regions $\mathcal{W}$, took on the order of 8 hours. In addition to the large amount of time to synthesize all of the individual controllers, numerous memory issues came up throughout the process, even with a system limit of 16 GB of RAM. The total size of the synthesized controllers was approximately 2 GB.

For each scenario tested, simulations where conducted 100 times to assess the fleet of UAVs' ability to slow down the spread of the fire to the domain edges, provided each UAV experiences a 1% chance of a random stop signal for every transition time (3 seconds in all scenarios). The results were compiled and displayed in box plots shown

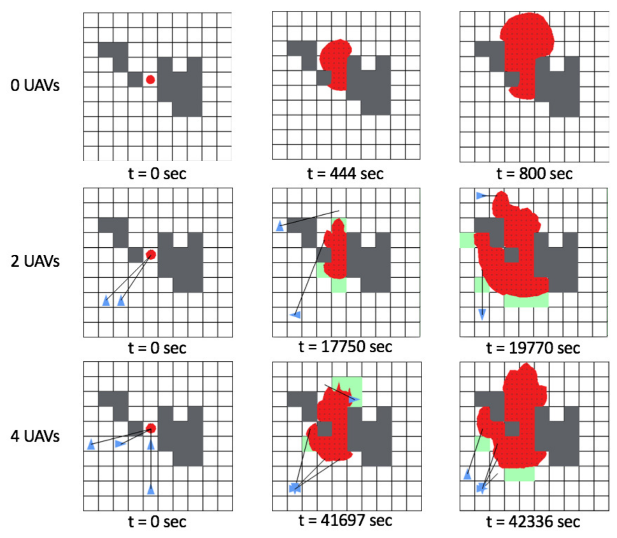


FIGURE 12. Example of scenario 1 results for no UAVs, 2 UAVs, and 4 UAVs. Red areas represent fire, blue triangles are the UAVs, black lines show the current UAV assignments, green squares are fire perimeter locations in which suppressant is currently dropped, and black squares represent the obstacles.

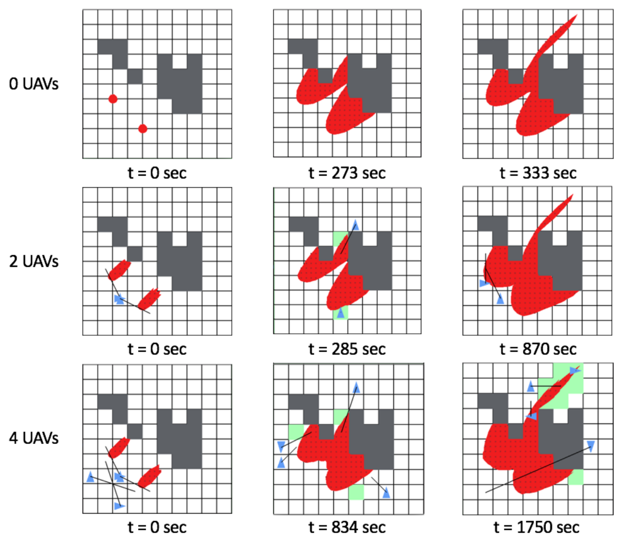


FIGURE 13. Example of scenario 4 results for no UAVs, 2 UAVs, and 4 UAVs.

in Fig. 10 and 11. The top and bottom of the boxes indicate the 75th and 25th percentiles, respectively, with red plus signs showing outliers. Median duration times of a distribution are represented by the middle red lines in the boxes, and mean duration times are shown as the asterisks alongside numerical values.

Additionally, example time lapses for Scenarios 1, 4, and 6 are presented in Fig. 12, 13 and 14, showcasing how the fire grew in response to a varying number of UAVs.

A few outcomes are observable through Fig. 10 and 11. First, in all cases, increasing the number of UAVs generally increased the median and average duration times of the tests, an intuitive result. Additionally though, especially evident

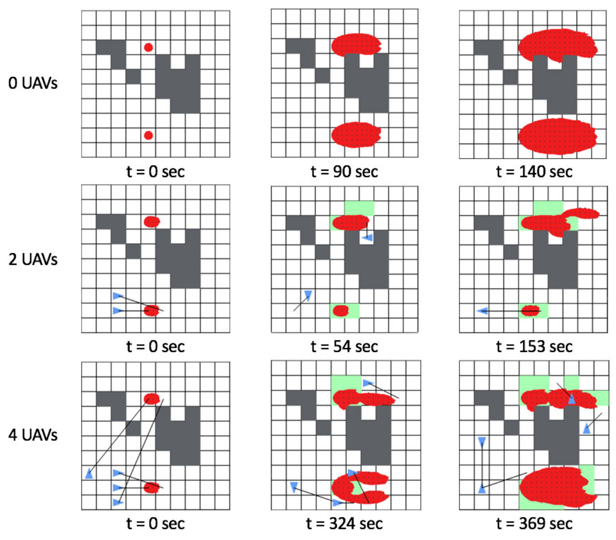


FIGURE 14. Example of scenario 6 results for no UAVs, 2 UAVs, and 4 UAVs.

in the Scenario 1 side of Fig. 10, the greater the number of UAVs used resulted in a higher spread between the minimum and maximum test duration times. The most likely explanation for this behavior is that greater differences between fire conditions in separate simulations accumulate over longer run-times associated with larger UAV groups, and since the UAVs are suspect to 4 minute periods of stopping while the standard 33% of 125 liters suppressant dropped corresponds to 3.33 minutes of burn-through time, these differences can greatly effect the UAVs' ability to slow down critical fires in time before refilling.

The greater stress scenarios in 4 and 6 (greater wind and number of fires) provide notable results in contrast to one another. In scenario 4, the obstacles provide additional blockage for the UAVs, and as a result, a greater number of UAVs provides greater performance since the number of critical fire locations (e.g. fires further in the wind direction and closer to the edge) are limited and easily accessible in time. This is evident in Fig. 13 in the 285 second time of the 2 UAV case. By only hitting the edges of the fire about to wrap around the obstacle, the fire was greatly slowed down. On the other hand, in scenario 6, few obstacles slowed the fires down, and the UAVs had to "rush" in time to suppress the fires. Multiple UAVs were always required for the test to have any chance of lasting longer than the 0 UAV case, but often UAVs could not reach the critical fires in time, evident in both the 2 UAVs and 4 UAVs cases of Fig. 14 and by the minimum duration values in the scenario 6 side of Fig. 11.

VII. CONCLUSION AND FUTURE WORK

In this paper, we constructed a high-level planner and controller to control a fleet of UAVs for various fire fighting scenarios. Our contributions include the RH framework modification combined with dynamic allocation, the algorithm for modifying the horizon definitions during synthesis, and the implemented simulation and results. The simulation demonstrated the method's ability to slow down the advancement of fire fronts towards the domain edge, providing a starting point of guaging the usefulness of automated UAVs in tackling fires before crews can arrive. The ability to slow down a starting fire by even half an hour to an hour (comparable to the maximum slowdown amounts in Scenarios 1, 2, 4, and 5) is a significant amount of time for ground crews to reach a location fast enough to effectively stop such a fire in its early phases.

Expanding upon the receding horizon framework for reactive synthesis allowed us to expand the scope of this problem while integrating the method with dynamic allocation for assigning UAVs. Even with such an approach, numerous issues arose throughout the process that help highlight key difficulties moving forward when using reactive synthesis in the control of UAVs. First, the RH framework, when considering all initial conditions, still yields an excessively large controller (about 2 GB) after 8 hours of runtime, a significant hurdle for applying such a design when considering arbitrary obstacle environments. Next, a simplified transition system was utilized which limited the total orientation space and interpreted UAV movement in only 2 dimensions, still far more restrictive than UAV movement in reality. Lastly, no constraints were used in considering the orientation of UAVs when dropping suppressant, which significantly factors into how well the suppressant slows down an advancing fire front. Each of these points combine to exemplify the need for smarter partitioning of possible transitions a UAV can take in 3D space (easily dependent on at least 3 full degrees of freedom), should reactive synthesize be used for UAV control. So while the reactive synthesis design is strong in enforcing the design constraints formally, the scope of its application is still limited per goal.

For improvements on this problem as it was explored, multiple changes can be assessed. First, the size of the synthesized controller could be addressed through reformatting the outputted synthesized controller in each horizon. Currently, controllers are synthesized per initial condition in a horizon, but synthesizing a single controller for each horizon that includes all initial conditions can cut down on the total size of the generated finite state machine and possibly the synthesis time. Second, modifications to the synthesized controllers can be made to enforce desired orientations during suppressant drops. This will intuitively add to the size of the controllers but enable more accurate control of the UAVs for dropping suppressant in the correct direction. Lastly, direct coordination between synthesized controllers should be explored to control the frequency of suppressant drops on critical fires. The need for such is apparent by the "escaping" streak of fire present for all cases in Fig. 13. If the UAVs had spread out the times they dropped suppressant on the fire wrapping around the obstacle edge, the advancement of such would be hindered further since the UAVs would avoid refilling at the same time and better stretch out their resources. This effect could be achieved through modifying the allocation algorithm to optimize the total assignment of UAVs through a finite time horizon, perhaps achieved by expanding on the method presented in [12] and performing a type of tree search across sequential UAV assignment options and their associated costs.

REFERENCES

[1] T. Wongpiromsarn, U. Topcu, and R. M. Murray, "Receding horizon temporal logic planning," *IEEE Trans. Autom. Control*, vol. 57, no. 11, pp. 2817–2830, Nov. 2012.

[2] D. Thomas, D. Butry, S. Gilbert, D. Webb, and J. Fung, "The costs and losses of wildfires: A literature survey," NIST Special Publication, Gaithersburg, MD, USA, Tech. Rep. 1215, Nov. 2017.

[3] M. P. Plucinski *et al.* "The effectiveness and efficiency of aerial firefighting in Australia. Part 1," Bushfire Cooperat. Res. Centre, Melbourne, VIC, Australia, Tech. Rep. A0701, 2006.

[4] D. Werner, "Fire drones," *Aerosp. Amer.*, vol. 53, no. 6, pp. 28–31, 2015.

[5] L. Merino, F. Caballero, J. R. Martínez-de Dios, I. Maza, and A. Ollero, "An unmanned aircraft system for automatic forest fire monitoring and measurement," *J. Intell. Robotic Syst.*, vol. 65, nos. 1–4, pp. 533–548, Jan. 2012, doi: 10.1007/s10846-011-9560-x.

[6] H. Qin *et al.*, "Design and implementation of an unmanned aerial vehicle for autonomous firefighting missions," in *Proc. 12th IEEE Int. Conf. Control Autom. (ICCA)*, Jun. 2016, pp. 62–67.

[7] L. Martin. (Dec. 2015). *Lockheed Martin Conducts Collaborative Unmanned Systems Demonstration*. [Online]. Available: https://news.lockheedmartin.com/2015-12-02-Lockheed-Martin-Conducts-Collaborative-Unmanned-Systems-Demonstration

[8] B. Johnson, F. Havlak, H. Kress-Gazit, and M. Campbell, "Experimental evaluation and formal analysis of high-level tasks with dynamic obstacle anticipation on a full-sized autonomous vehicle," *J. Field Robot.*, vol. 34, no. 5, pp. 897–911, 2017. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/rob.21695

[9] K. W. Wong and H. Kress-Gazit, "Need-based coordination for decentralized high-level robot control," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2016, pp. 2209–2216.

[10] J. Alonso-Mora, J. A. DeCastro, V. Raman, D. Rus, and H. Kress-Gazit, "Reactive mission and motion planning with deadlock resolution avoiding dynamic obstacles," *Auton. Robots*, vol. 42, no. 4, pp. 801–824, Apr. 2018, doi: 10.1007/s10514-017-9665-6.

[11] B. Mirzapour and H. S. Aghdasi, "Modified multi-objective hybrid DE and PSO algorithms for resource allocation in forest fires," in *Proc. 7th Int. Conf. Comput. Knowl. Eng. (ICCKE)*, Oct. 2017, pp. 187–192.

[12] J. D. Griffith, M. J. Kochenderfer, R. J. Moss, V. V. Misic, V. Gupta, and D. Bertsimas, "Automated dynamic resource allocation for wildfire suppression," *Lincoln Lab. J.*, vol. 22, no. 2, pp. 38–59, 2017.

[13] J. Tumova and D. V. Dimarogonas, "Multi-agent planning under local LTL specifications and event-based synchronization," *Automatica*, vol. 70, pp. 239–248, Aug. 2016. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0005109816301285

[14] N. Piterman, A. Pnueli, and Y. Sa'ar, "Synthesis of reactive(1) designs," in *Verification, Model Checking, and Abstract Interpretation*, E. A. Emerson and K. S. Namjoshi, Eds. Berlin, Germany: Springer, 2006, pp. 364–380.

[15] A. Ulusoy and C. Belta, "Receding horizon temporal logic control in dynamic environments," *Int. J. Robot. Res.*, vol. 33, no. 12, pp. 1593–1607, 2014, doi: 10.1177/0278364914537008.

[16] X. C. Ding, M. Lazar, and C. Belta, "Receding horizon temporal logic control for finite deterministic systems," in *Proc. Amer. Control Conf. (ACC)*, Jun. 2012, pp. 715–720.

[17] M. A. Finney, "FARSITE: Fire area simulator—Model development and evaluation," U.S. Dept. Agricult., Forest Service, Rocky Mountain Res. Station, Ogden, UT, USA, Res. Paper RMRS-RP-4, 2004, p. 47.

[18] P. L. Andrews, "The Rothermel surface fire spread model and associated developments: A comprehensive explanation," U.S. Dept. Agricult., Forest Service, Rocky Mountain Res. Station, Fort Collins, CO, USA, Tech. Rep. RMRS-GTR-371, Mar. 2018.

[19] J. S. Gould *et al.*, "Assessment of the effectiveness and environmental risk of the use of retardants to assist in wildfire control in Victoria," Austral. Dept. Natural Resour. Environ., CSIRO, Canberra, SW, Australia, Tech. Rep. 50, Feb. 2000.

[20] T. Wongpiromsarn, U. Topcu, N. Ozay, H. Xu, and R. M. Murray, "TuLiP: A software toolbox for receding horizon temporal logic planning," in *Proc. 14th Int. Conf. Hybrid Syst., Comput. Control*, New York, NY, USA, 2011, pp. 313–314, doi: 10.1145/1967701.1967747.

JOSHUA A. SHAFFER received the B.S. degree in aerospace engineering from Pennsylvania State University, State College, PA, USA, in the spring of 2017. He is currently pursuing the M.S. degree in aerospace engineering with the University of Maryland (UMD) at College Park, College Park, MD, USA.

He is currently a Research Assistant with the Department of Aerospace Engineering, UMD, under the supervision of Dr. H. Xu. His research topics involve the implementation of formal methods in dynamic path planning for systems involving UAVs.

ESTEFANY CARRILLO received the B.S. and M.S. degrees in electrical engineering from the University of Maryland at College Park, College Park, MD, USA, in 2012 and 2017, respectively, where she is currently pursuing the Ph.D. degree in aerospace engineering. Her research focuses on the use of formal methods and hybrid systems theory in the design of verifiable controllers for complex high-level tasks and control of multi-agent systems.

HUAN XU received the B.S. degree in mechanical engineering and material science from Harvard University, in 2007, and the M.S. and Ph.D. degrees in mechanical engineering from the California Institute of Technology, in 2008 and 2013, respectively. She is currently an Assistant Professor with the Department of Aerospace Engineering, Institute for Systems Research, University of Maryland at College Park, College Park. Her work focuses on the use of formal methods and controls in the design and analysis of unmanned autonomous systems.