A Competitive Algorithm for Online Multi-Robot Exploration of a Translating Plume

Yoonchang Sung and Pratap Tokekar

***Abstract*— In this paper, we study the problem of exploring a translating plume with a team of aerial robots. The shape and the size of the plume are unknown to the robots. The objective is to find a tour for each robot such that they collectively explore the plume. Specifically, the tours must be such that each point in the plume must be visible from the field-of-view of some robot along its tour. We propose a recursive Depth-First Search (DFS)-based algorithm that yields a constant competitive ratio for the exploration problem. The competitive ratio is $\frac{2(S_r+S_p)(R+\lfloor \log R \rfloor)}{(S_r-S_p)(1+\lfloor \log R \rfloor)}$ where R is the number of robots, and S_r and S_p are the robot speed and the plume speed, respectively. We also consider a more realistic scenario where the plume shape is not restricted to grid cells but an arbitrary shape. We show our algorithm has $\frac{2(S_r+S_p)(18R+\lfloor \log R \rfloor)}{(S_r-S_p)(1+\lfloor \log R \rfloor)}$ competitive ratio under the *fat* condition. We empirically verify our algorithm using simulations.**

I. Introduction

We investigate the problem of exploring and mapping flows of an unknown hazardous agent in aquatic environments using a team of autonomous aerial robots. Our overall vision is to develop algorithms for enabling a team of robots to assist emergency responders in disaster scenarios, such as dispersal of oil aerosols and radioactive particulates in the environment. limiting their ability to respond quickly and effectively. This motivates the use of Unmanned Aerial Vehicles (UAVs) which can provide a wider (regional) picture and that can coordinate with USVs for more targeted deployments.

Fig. 1. An aerial robot (UAV) conducting the plume exploration in an abandoned quarry near Blacksburg, Virginia.

Teams of UAVs can collectively track the plumes and act as scouts to direct the USVs to sense for hazardous regions of interest (Figure 1). As a first step towards enabling coordination between UAVs and USVs, in this paper, we focus on the problem of mapping the extent of a 2D plume.

The problem of exploring an unknown 2D environment is a well-studied one in the robotics [1]–[4] and computational geometry [5]–[8] communities. The problem considered in this paper differs from these works in two critical ways. First, we consider the case where the plume is not static but is instead translating with a given velocity. As a result, the performance of the algorithm depends on the relative speeds of the robots and the plume. Second, in our setup the robots are not restricted to stay inside (or over) the plume all the time. The robots can fly over locations that are not part of the plume, thereby allowing them to "shortcut" from one part of the plume to the other. Contrast this with conventional 2D exploration problems, where the robots are restricted to stay within the boundary of the environment. Because the robots do not know the shape of the plume a priori, they may not be able to take a "shortcut" even if one exists. As a result, the robots may ending up taking a longer path, resulting in a poorer performance. Nevertheless, we present an algorithm that is *competitive* with respect to the optimal algorithm.

We use the notion of *competitive ratio* [9] to analyze the performance of our algorithm. The competitive ratio for an online algorithm is defined as the largest (i.e., worst-case input) ratio of the time taken by the online algorithm to the time taken by an offline, optimal algorithm. The offline algorithm is one which knows the shape of the 2D plume a priori. We seek algorithms that have a low (preferably, constant) competitive ratio. Our main result is a constant competitive ratio for exploring a translating plume for a fixed number of robots. The constant depends on the relative speeds of the plume and the robots.

We require the robots to ensure that all points of the plume are within the Field-of-View (FOV) of at least one of the aerial robots along their paths. The objective is to minimize the time required for all the robots to explore the plume and return back to the starting position. Our algorithm builds on the one presented by Higashikawa *et al*. [8] for exploring an unknown binary tree. We show how to reduce the problem of exploring the plume to that of exploring a binary tree. We first start with the simpler scenario where the plume is modeled as a 2D grid and then generalize it to the case where the plume boundary is any smooth (formally defined in Section II 2D curve. For both cases, we show that our algorithm yields a constant-competitive ratio.

Related Work: The problem of monitoring an environmental monitoring has been extensively studied in robotics [10] because of its relevance to a number of applications. These applications include precision agriculture [11], [12], wildlife habitat monitoring [13]–[15] and atmospheric plume tracking [16]–[18]. A basic capability for environmental monitoring is that of area exploration and coverage. Galceran and Carreras [19] survey planning algorithms for area coverage under a variety of sensing and motion models.

*This material is based upon work supported by the National Science Foundation under Grant No. 1637915.

The authors are with the Department of Electrical and Computer Engineering, Virginia Tech, USA. {yooncs8, tokekar}@vt.edu.

The first step in our algorithm is to model the plume as a 2D grid polygon. Exploring an unknown grid polygon has been studied under two models: *lawn mowing* and *milling*. The former allows a robot to move outside the boundary of the polygon whereas the latter does not and restricts the motion of the robot to always be inside the polygon. Arkin *et al*. [20] presented $(3+\epsilon)$–approximation algorithm for offline lawn mowing and 2.5–approximation algorithm for offline milling with a single robot. Arya *et al*. [21] presented an approximation algorithm for offline milling for multiple robots. Here, offline denotes the fact that the algorithm knows the polygon a priori. The problem we consider is that of online lawn mowing.

Icking and Kamphans [22] proposed a strategy that yields a competitive tour for online milling of an polygon which may contain holes with a single robot. Icking [5] presented a $\frac{4}{3}$–competitive algorithm for online milling of polygons without holes. Kolenderska *et al*. [6] improved upon this to give an online milling algorithm with a competitive ratio of $\frac{5}{4}$. None of these online algorithms are designed for multiple robots. Furthermore, in all of these works, the grid environment is static. We present the first constant-competitive ratio algorithm for online lawn mowing with multiple robots.

The objective of online exploration strategies [2]–[4], [23] is to explore and map an unknown environment. When multiple robots are considered, a common strategy [1], [7], [8], [24] is to abstract the problem as that of exploring a tree by employing a recursive Depth-First Search (DFS). Fraigniaud [7] proposed $\mathcal{O}(R\log R)$–competitive algorithm where R denotes the number of robots. Brass *et al*. [1] and Higashikawa *et al*. [8] improved this competitive ratio to $\frac{2e}{R}+\mathcal{O}((R+r)^{R-1})$ where e denotes the number of edges and r is the radius of graph and to $\frac{R+\lfloor\log R\rfloor}{1+\lfloor\log R\rfloor}$, respectively.

In these works, the environment to be explored is assumed to be a tree. Preshant *et al*. [24] showed that the competitive ratio remains largely the same, $\frac{2(\sqrt{2}R+\log R)}{1+\log R}$, when the environment is an orthogonal polygon[1] but is modeled as a tree. We build on this and generalize this to the case where the environment boundary is not necessarily orthogonal. In fact, it can be curved and may contain holes as well. Furthermore, we show how to adapt this algorithm to the case where the environment itself is translating.

The contributions of this paper are as follows:

- If the plume is a 2D grid polygon, then we present an online exploration algorithm for R robots with a competitive ratio of $\frac{2(S_r+S_p)(R+\lfloor\log R\rfloor)}{(S_r-S_p)(1+\lfloor\log R\rfloor)}$.
- If the plume has an arbitrary shape, then our online exploration algorithm yields a competitive ratio of $\frac{2(S_r+S_p)(18R+\lfloor\log R\rfloor)}{(S_r-S_p)(1+\lfloor\log R\rfloor)}$.

Here S_r and S_p are the speeds of the robots and the plume, respectively.

[1]An orthogonal polygon is one in which the edges are aligned with either the X or Y axes.

II. Problem Description

We consider the problem of mapping a slowly translating plume (Definition 1) using a team with R robots. The size and the shape of the plume is not known to the robots a priori. We use $P\in\mathcal{R}^2$ to denote the 2D plume. Let $int(P)$ be the interior of P and ∂P be the boundary of P.

We assume the plume is translating on the plane at zero height and that the aerial robots fly at a fixed altitude. Each robot has a downwards-facing camera that yields a square footprint on the plane containing the plume. That is the FOV is a square. Without loss of generality, we assume that the side length of the square FOV is 1 in this work.

We consider plumes whose size of the plume is at least as large as the FOV of the robots. Specifically, we require the plume to satisfy the following assumption.

Definition 1 (Fat Plumes)**.** *For any $p'\in\partial P$, let $p\in int(P)$ be a point on the normal to ∂P at p' such that p is at a distance of $\frac{\sqrt{2}}{2}$ from p'. Let $B(p)$ be an open ball of radius $\frac{\sqrt{2}}{2}$, i.e., $B(p)=\{q\mid \|p-q\|_2<\frac{\sqrt{2}}{2}\}$ where $q\in\mathcal{R}^2$. We say that the plume P is* fat *if $B(p)$ lies completely inside $int(P)$ for all $p'\in\partial P$.*

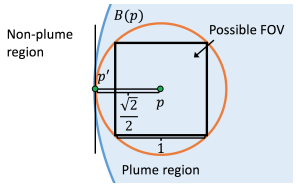


Fig. 2. We restrict our attention to plumes that are *fat* (Definition 1).

Figure 2 shows an example of a plume that is *fat*. This definition disallows plumes that have a *width* less than that of the FOV of the robot. Note, however, we still allow the plume to contain one or more holes.

We assume that the plume translates at a fixed velocity of S_p which is known to the robots. [2] The velocity of the plume can be determined from the flow of the water which can be found from the environmental conditions such as wind and ocean current models [25].

We assume that all robots move at a speed of $S_r>S_p$. We further assume that all robots can communicate with each other at all times and thus, restrict our attention to centralized algorithms.

We focus on the mapping problem in this paper. Therefore, we assume that all robots start at the same location where they first observe the plume. We seek tours for each robot that explore the plume and return back to this starting location.

Problem 1 (Multi-Robot Exploration of Translating Plume)**.** *Find a tour for all the robots that minimizes the exploration time such that every point in the plume is visible from the FOV of at least one robot's tour. All tours must return to the same starting position. The exploration time is given by the time when the last robot returns to the starting position.*

The proposed problem is an online exploration problem. The objective function is the exploration time which is the

[2]This is equivalent to the rigid-body translation of P.

time of the longest tour. In the next section, we present an algorithm that is based on recursive DFS which is competitive with respect to the optimal solution.

III. Plume Exploration Algorithm

In this section, we present our main algorithm. We first solve a simpler version of Problem 1 where the plume is approximated as a grid map. We then use this result to solve Problem 1 by relaxing the grid approximation afterwards. Our algorithm is based on the recursive DFS that models the plume under exploration as a tree. We first show that our strategy is competitive for the grid map case and then analyze the effect of approximating an arbitrary plume shape with a grid.

A. Recursive DFS Algorithm for a Grid Map

In this section, we assume that the plume is represented as a grid map [26]. The environment is modeled as a collection of cells, each of which is a square of unit side length. Each cell is connected to four of its neighbors. The plume P is just a collection of C cells that form one connected set (if a cell $c \in P$ is part of the plume, then one of its four neighbors must also be a part of the plume when $C > 1$).

The problem of exploring the plume is then simplified to that of exploring a grid map and identify the cells that belong in P. Since we assume that the FOV is also a unit square, a robot may obtain an image by positioning itself at the center of a cell. By analyzing the pixels on the boundary of the image, the robot can then determine if any of the four neighboring cells are also part of the plume or not.

We model P as a tree and propose a recursive DFS algorithm based on the tree exploration algorithm given by Higashikawa *et al*. [8]. Higashikawa *et al*. [8] developed a recursive DFS algorithm for exploring a binary tree. In our case, the grid graph to be explored is not necessarily a tree (it may contain cycles). Regardless, we show that modeling the underlying graph as a binary tree still leads to an algorithm with a constant competitive ratio.

The root of the tree is the cell corresponding to the starting position of the robots. Upon visiting a cell, the robots can identify if one or more of the four neighboring cells also contain the plume. The neighboring cells that contain the plume are added as children of the present cell in the tree unless those cells have been previously added to the tree. This condition prevents cycles.

The number of neighboring cells when a robot visits a new cell can be at most three. Therefore, the resulting tree may not be binary. However, by introducing a dummy edge of weight 0, we can convert the tree into a binary tree without loss of generality.[3]

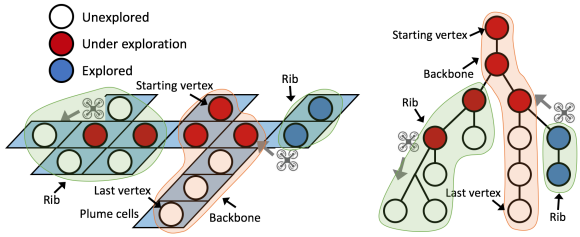
Each neighboring plume cell determined by the sensing model becomes one of candidate cells that robots can choose from as the next vertex to visit. The goal becomes to visit all $C-1$ cells (excluding the starting cell) at least once by one of the robots.

[3]This step is included in Line 15 of Algorithm 1.

The weight of an edge is equal to the time taken by the robot to go from the center of one cell to that of the neighbor. Since we know the velocity with which the plume is translating, we can determine the location of the center of the neighboring cell at a future time instance using the relative velocities between the plume and the robots.

If $R = 1$, then our algorithm becomes conventional recursive DFS for a single robot. However, in the multi-robot case, as the robots build the tree, we split the robots as equally as possible and assign them to explore the children vertices.

We define three states for each vertex in the tree: *unexplored* if the vertex is not visited by any robots; *under exploration* if the vertex is visited by any robots but the leaf vertex connected from the vertex is not visited by any robots; and *explored* if the vertex as well as the leaf vertex in the same branch are visited by any robots. When robots decide which vertex to move among neighboring cells of a plume region, they do not consider *explored* vertices but vertices that are either *unexplored* or *under exploration*. This is because having *explored* vertex means that the offspring of it must have also been explored by any robots (see Figure 3).



(a) Two robots exploring the grid map. (b) Tree generated from the recursive DFS.

Fig. 3. Description of tree components. The binary tree consists of a backbone and a finite number of ribs. Each vertex is marked as one of *unexplored*, *under exploration* or *explored*.

The details are given in Algorithm 1[4]. All vertices are marked as *unexplored* state in the beginning. Each robot runs Algorithm 1 whenever it reaches a vertex. The algorithm can be implemented to a single robot independently with respect to other robots as long as they can share the state information of vertices. The algorithm terminates when all robots return to the starting vertex and all vertices are marked as *explored*.

B. Theoretical Analysis

In this section we analyze the proposed Algorithm 1. We start with the following lemma which bounds the total weight of the tree generated by the robots.

Lemma 1 (Total Weight of the DFS Tree)**.** *Let L be the sum of the weights of all the edges in the tree generated by the recursive DFS algorithm. We have $L \leq \frac{C-1}{S_r - S_p}$.*

Proof. A tree containing C vertices always has $C-1$ edges. The weight of an edge is equal to the time taken by the robot to go between the centers of the respective cells. The

[4]In the algorithm, we use $N(v_i)$ to denote the neighborhood of the i-th vertex such that $N(v_i) = \{v_j \in V | (v_j, v_i) \in E\}$.

Algorithm 1: Multi-Robot Recursive DFS

1 Observe $N(v)$ to determine whether neighboring cells are plume cells or non-plume cells.
2 **if** $|N(v)|$=*0* **then**
3 | Mark v as *explored*.
4 | Move back to the parent vertex ($\rightarrow$next vertex) and directly jump to Line 24.
5 **end**
6 Communicate with robots to update the state of $N(v)$, *i.e.*, *unexplored*, *under exploration*, and *explored*.
7 $N(v) \leftarrow N(v) \backslash \{$*explored* vertices$\}$.
8 **if** $v' \in N(v)$ *is under exploration* **then**
9 | **if** *moving to* v' *generates a cycle in the tree* **then**
10 | | $N(v) \leftarrow N(v) \backslash \{v'\}$.
11 | **end**
12 **end**
13 **if** $|N(v)| > 1$ **then**
14 | **if** $|N(v)| > 2$ **then**
15 | | Add a dummy edge of weight 0 in order to keep the tree as a binary tree.
16 | **end**
17 | Split robots into two children as equally as possible.
18 | Move to one of two children ($\rightarrow$next vertex) and mark v as *under exploration*.
19 **else if** $|N(v)| = 1$ **then**
20 | Move to the child ($\rightarrow$next vertex) and mark v as *under exploration*.
21 **else if** $|N(v)| = 0$ **then**
22 | Move back to the parent vertex ($\rightarrow$next vertex).
23 **end**
24 $v \leftarrow$the next vertex.

cells are a unit distance apart from each other. However, the robot travels with a speed of S_r and the plume (thereby, the center of the cell) travels with a speed of S_p. The worst-case time taken by the robot to travel a unit distance occurs when the plume (i.e, the center) is moving directly away from the robot. The minimum relative speed is $S_r - S_p$. Therefore, the maximum time taken for executing an edge is $\frac{1}{S_r - S_p}$. Since there are $C-1$ edges in the tree, the total weight will be less than $\frac{C-1}{S_r - S_p}$. □

a) Upper Bound Analysis: To analyze the cost of the proposed algorithm, we adapt the token collecting rule proposed by Higashikawa *et al*. [8] to the case of a translating plume. They place tokens with a total weight of $2(L - d_{max}) + (1 + \lfloor \log R \rfloor) d_{max}$ spread over the edges. Here, d_{max} denotes the distance of the farthest vertex in the tree from the root. The details of the token collecting rules are given in Corollary 6 in [8]. They assume that the robots move at unit speed. We only describe how the rule is adapted to handle the case of a translating plume (i.e., tree). The reader is referred to the Corollary 6 in [8] for the full details.

The modified rule is as follows: (1) At least $(S_r - S_p)$ and at most $(S_r + S_p)$ tokens per unit time are collected by one robot in a group that visits an edge e in a rib in the forward direction for the first time. (2) At least $(S_r - S_p)$ and at most $(S_r + S_p)$ tokens per unit time are collected by one robot in a group that visits an edge e in a rib in the backward direction for the first time. (3) At least $(S_r - S_p)$ and at most $(S_r + S_p)$ tokens per unit time are collected by each of $1 + \lfloor \log R \rfloor$ robots that move along a backbone edge e in the forward direction for the first time. (4) At least $(S_r - S_p)$ and at most $(S_r + S_p)$ tokens per unit time are collected by one robot in a group that move along a backbone edge e in the forward direction after the first group.

Let t_{last} be the time when the last robot reaches a leaf node in the tree. Then, based on the four observations above we have:

$$\begin{aligned}(S_r - S_p)(1 + \lfloor \log R \rfloor) t_{last} \leq \textit{total amount of tokens} \\ \leq (S_r + S_p)(1 + \lfloor \log R \rfloor) t_{last}.\end{aligned} \tag{1}$$

The *total amount of tokens* is equal to $2(L - d_{max}) + (1 + \lfloor \log R \rfloor) d_{max}$.

We denote the time taken by the proposed algorithm by ALG. We are now ready to state the upper bound on ALG.

Lemma 2 (Upper Bound for Multi-Robot Recursive DFS)**.**

$$ALG \leq \frac{2(C + d_{max} \lfloor \log R \rfloor)}{(S_r - S_p)(1 + \lfloor \log R \rfloor)}. \tag{2}$$

The proof is given in Appendix A.

Corollary 1 (Special Cases)**.** *Upper bounds for the following special cases can be derived from Lemma 2, such as Multi-Robot Static Plume (MRSP), Single Robot Translating Plume (SRTP), and Single Robot Static Plume (SRSP).*

MRSP	SRTP	SRSP
$\texttt{ALG} \leq \frac{2(C + d_{max} \lfloor \log R \rfloor)}{S_r (1 + \lfloor \log R \rfloor)}$	$\texttt{ALG} \leq \frac{2C}{S_r - S_p}$	$\texttt{ALG} \leq \frac{2C}{S_r}$

TABLE I
UPPER BOUNDS OF SPECIAL CASES.

Note that the upper bound for MRSP becomes the result from Higashikawa et al. [8] if $S_r = 1$. Also, the upper bound for SRSP is equivalent to Icking et al. [22] if $S_r = 1$.

Proof. Please refer to Appendix B. □

b) Lower Bound Analysis: We study the lower bound for the optimal algorithm in order to obtain a competitive ratio. Let $\texttt{OPT}_g^1$ be the time taken by the optimal algorithm to explore a grid map when using a single robot. The lower bound can be constructed as:

$$\texttt{OPT}_g^1 \geq \frac{C-1}{S_r + S_p}. \tag{3}$$

We use $\texttt{OPT}_g^R$ to represent the time taken by the optimal algorithm over any grid polygon of a plume region using R robots. Then, the following lemma gives the lower bound for $\texttt{OPT}_g^R$.

Lemma 3 (Lower Bound for Optimal Algorithm)**.**

$$OPT_g^R \geq \frac{C-1}{(S_r + S_p)R}. \tag{4}$$

Proof. Please refer to Appendix C. □

Theorem 1 (Competitive Ratio over the Grid Polygon)**.** *The competitive ratio of Algorithm 1 for a grid map is:*

$$\begin{aligned} ALG \leq & \frac{2(S_r+S_p)(R+\lfloor \log R \rfloor)}{(S_r-S_p)(1+\lfloor \log R \rfloor)} OPT_g^R \\ & + \frac{2}{(S_r-S_p)(1+\lfloor \log R \rfloor)}. \end{aligned} \tag{5}$$

Proof. Substituting Equation (4) into Equation (2) gives:

$$\texttt{ALG} \leq \frac{2((S_r+S_p)R\texttt{OPT}_g^R + 1 + d_{max}\lfloor \log R \rfloor)}{(S_r-S_p)(1+\lfloor \log R \rfloor)}. \tag{6}$$

Since $\frac{d_{max}}{S_r+S_p} \leq \texttt{OPT}_g^R$, it follows:

$$\leq \frac{2(S_r+S_p)(R+\lfloor \log R \rfloor)\texttt{OPT}_g^R + 2}{(S_r-S_p)(1+\lfloor \log R \rfloor)}. \tag{7}$$

□

C. Arbitrary Plume Shape

The presented results so far are for a grid map approximation of the plume. In this section, we will relate the bounds obtained for the grid map case to the case of arbitrarily-shaped plumes. Specifically, we will extend Lemma 2 to apply to a plume region that may have an arbitrary shape.

The algorithm for exploring the plume remains the same. We will still construct a tree that represents a grid map of the plume. The main difference here is that in the previous analysis, we assumed that the boundary of the plume matched the boundary of a grid map exactly. This will no longer hold. Instead, we will explore a grid map that is an *outer* approximation of the plume (Figure 4).

We define C_{out}^{ALG} and C_{in}^{ALG} to denote the number of cells in the outer and inner grid approximation by our algorithm, respectively. The outer grid map completely contains the plume whereas the inner grid map lies completely inside the plume. Therefore, the term C in the upper bound (Lemma 2) will now be replaced by C_{out}^{ALG}. However, the C term in the lower bound (Lemma 3) cannot be replaced by C_{in}^{ALG}. This is because C_{in}^{ALG} is defined by the grid imposed by our algorithm. It may be possible to have another grid map (of the same unit side length) that is oriented and/or translated such that it contains fewer than C_{in}^{ALG} cells in the interior. We will first find the relationship between C_{out}^{ALG} and C_{in}^{ALG}. Then, we will relate C_{in}^{ALG} to C_{in}^{BEST} which is the best grid that contains the fewest number of cells completely inside the plume.

By a slight abuse of notation, we interchangeably use C_{out}^{ALG} and C_{in}^{ALG} to also denote the corresponding set of cells (along with denoting the number of cells in the set).

Lemma 4 (Grid Approximation of Arbitrary Plume Shape)**.** *The upper bound on C_{out}^{ALG} for a* fat *polygon (from Definition 1) is given by:*

$$C_{out}^{ALG} \leq 3C_{in}^{ALG} + 6. \tag{8}$$

Proof. To prove the lemma, we define an EXCESS set that contains all cells, $p \in P \backslash C_{in}^{ALG}$. That is, EXCESS set contains all cells in C_{in}^{ALG} but not in C_{in}^{ALG}. Therefore, the size of the EXCESS set is equal to $C_{in}^{ALG} - C_{in}^{ALG}$. We prove the lemma in three steps.

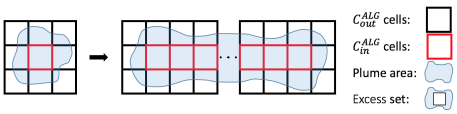


Fig. 4. Row formation of C_{in}^{ALG} cells as the number of cells changes from 1 to a finite number.

EXCESS is maximum if all cells in C_{in}^{ALG} form a convex polygon. If there is a reflex vertex vertex in C_{in}^{ALG}, then the reflex vertex cell does not contribute any cell to the EXCESS set that is not already contributed by one of the neighbors of the reflex vertex. Since C_{in}^{ALG} is a grid map, the only convex shape possible is a rectangle.

If the width of the rectangle is equal to one, then each cell in C_{in}^{ALG} contributes two cells that are in the EXCESS set (one above and one below) in addition to three more cells on either end point. This is shown in Figure 5. Therefore, the size of EXCESS set is equal to $2C_{in}^{ALG} + 6$.

If the width of the rectangle is more than 1, then each cell will contribute at most one addition cell in the EXCESS set. Therefore, the size of the EXCESS set is less than or equal to $C_{in}^{ALG} + 8$.

The width of the rectangle cannot be less than 1; it violates the *fat* condition for the polygon. Therefore, the maximum possible value for the size of the EXCESS set is $2C_{in}^{ALG} + 6$. By substituting EXCESS with $C_{out}^{ALG} - C_{in}^{ALG}$, we have Equation (8). □

The grid corresponding to C_{our}^{ALG} and C_{in}^{ALG} is generated by the proposed algorithm. It is possible that there exists some other grid which has fewer than C_{in}^{ALG} cells completely contained within the plume. It may not be possible to generate this "best" grid due to the nature of online exploration. Nevertheless, we analyze the relationship between C_{in}^{ALG} and C_{in}^{BEST}. We define C_{in}^{BEST} to denote the fewest number of cells in the inner grid approximation that is completely contained in the plume (and adding any other cell to C_{in}^{BEST} would not allow C_{in}^{BEST} to be completely inside the plume). The relationship is given by:

Lemma 5 (Best Possible Grid-Approximation)**.**

$$C_{in}^{ALG} \leq 6C_{in}^{BEST}. \tag{9}$$

Proof. To prove this relationship, it is sufficient to consider any grid approximation (generated by any algorithm) with respect to the best grid approximation.

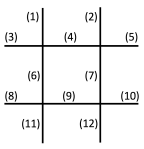


Fig. 5. A part of grid cell from any grid approximation. Unique number is assigned to a different side of grid cells.

Figure 5 shows a part of grid cells generated by any grid approximation. Each number in the figure corresponds to a different side of grid cells. Let c_{in}^{BEST} be a single grid cell

generated from the best grid approximation that overlaps with the central cell $(4, 6, 7, 9)$ without loss of generality. Our observation is that the number of crossings is equal to the number of cells in C_{in}^{ALG} that c_{in}^{BEST} overlaps.

We prove that 7 crossings are impossible. In order to cross more than four edges, c_{in}^{BEST} has to cross all of the (4, 6, 7, 9) edges. In addition, it must cross three of $(1, 2, 3, 5, 8, 10, 11, 12)$ edges. Let use consider the case when edge (1) is crossed. The other cases are symmetric. If edge (1) is crossed, then crossing (5, 10, 12, 11, 8) is impossible since these edges are more than a unit distance apart. Only (2) and (3) edges are only available edges to cross more. However, if c_{in}^{BEST} crosses both (2) and (3) edges, the side length of c_{in}^{BEST} becomes greater than 1. Therefore, c_{in}^{BEST} cannot cross more than seven edges. Therefore, $C_{in}^{ALG} \leq 6C_{in}^{BEST}$. □

Finally we give our main result as follows:

Theorem 2 (Competitive Ratio for Arbitrary Plume Shape)**.** *Let $\texttt{OPT}^R$ be the time taken by the optimal algorithm over any arbitrary plume shape using R robots.*

$$\begin{aligned} ALG \leq & \frac{2(S_r + S_p)(18R + \lfloor \log R \rfloor)}{(S_r - S_p)(1 + \lfloor \log R \rfloor)} OPT^R \\ & + \frac{48}{(S_r - S_p)(1 + \lfloor \log R \rfloor)}. \end{aligned} \tag{10}$$

Proof. Although $\texttt{OPT}^R$ is the cost for any arbitrary plume shape, we can still lower bound this using C_{in}^{BEST} (similar to Lemma 3) as:

$$\texttt{OPT}^R \geq \frac{C_{in}^{BEST} - 1}{(S_r + S_p)R}. \tag{11}$$

Let $(S_r - S_p)(1 + \lfloor \log R \rfloor)$ be $\mathcal{M}$. We can obtain the following inequalities from Lemmas 2, 4, and 5 as follows.

$$\texttt{ALG} \leq \alpha C_{out}^{ALG} + a \leq \beta C_{in}^{ALG} + b \leq \gamma C_{in}^{BEST} + b, \tag{12}$$

where $\alpha = \frac{2}{\mathcal{M}}$, $a = \frac{2d_{max}\lfloor \log R \rfloor}{\mathcal{M}}$, $\beta = \frac{6}{\mathcal{M}}$, $b = \frac{12 + 2d_{max}\lfloor \log R \rfloor}{\mathcal{M}}$, and $\gamma = \frac{36}{\mathcal{M}}$.

Substituting Equation (11) into the last inequality of Equation (12) and using $\frac{d_{max}}{S_r + S_p} \leq \texttt{OPT}^R$, we have:

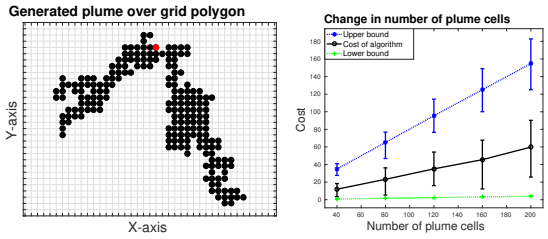
$$\begin{aligned} \texttt{ALG} &\leq \frac{36(S_r + S_p)R}{\mathcal{M}} \texttt{OPT}^R + \frac{48 + 2d_{max}\lfloor \log R \rfloor}{\mathcal{M}}, \\ &\leq \frac{2(S_r + S_p)(18R + \lfloor \log R \rfloor)}{\mathcal{M}} \texttt{OPT}^R + \frac{48}{\mathcal{M}}. \end{aligned} \tag{13}$$

□

IV. Simulation

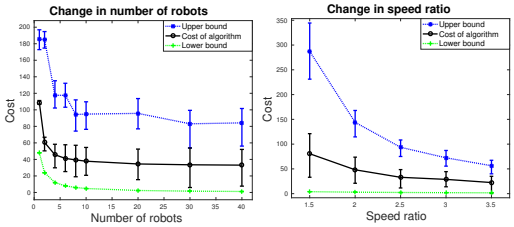
We empirically evaluated our algorithm using MATLAB simulations. Specifically, we verify the performance of the proposed recursive DFS for the grid map approximation of the plume (Theorem 1).

We randomly generated a set of plume grid maps. Figure 6 (a) shows an example of the generated plume that consists of 200 cells. We measured the cost of our algorithm as well as the upper and lower bounds by changing the number of plume cells, the number of robots, and the speed ratio between the robot and the plume. Each case was obtained from 100 trials (see Figures 6 (b–d)).



(a) Example of the randomly generated plume over grid cells. The red dot represents the starting vertex for robots.

(b) Plot of the cost when changing the number of plume cells.



(c) Plot of the cost when changing the number of robots.

(d) Plot of the cost when changing the speed ratio between the robot and the plume.

Fig. 6. Simulation results. We fixed the number of plume cells, the number of robots, the speed ratio as 120, 20, 2.5, respectively, when the corresponding variable was not a subject to be changed. We ran 100 trials for each case. Each case is plotted as mean, maximum and minimum values from 100 trials.

Figure 6 (b) shows that the expected exploration time for all cases is proportional to the number of plume cells. The difference between the maximum and minimum costs also becomes larger as more plume cells are to be explored. Figure 6 (c) plots the exploration time when changing the number of robots. The exploration time of our algorithm and the lower bound decrease as the number of robots increases. Unlike these, the upper bound does not show a steady decreasing tendency because randomly generated plume cells affect d_{max}. Figure 6 (d) shows the exploration time when changing the speed ratio between the robot and the plume, *i.e.*, $\frac{S_r}{S_p}$. The exploration time for our algorithm and the upper bound decrease as the speed ratio increases. The simulation results verify the theoretical upper and lower bounds determined by our analysis. In addition, they demonstrate that the practical performance of our algorithm is better than that indicated by the upper bounds.

V. Conclusion

We propose a recursive DFS algorithm for a team of aerial robots to explore a translating plume without knowing its shape and size. We present two approaches for the given problem where the first approximates the plume to map to the grid whereas the second considers any arbitrary shape of plume as long as it is *fat*. Both approaches are competitive with respect to the optimal algorithm. Immediate future work would be to verify the performance of our algorithm to the plume having any arbitrary shapes.

REFERENCES

[1] P. Brass, F. Cabrera-Mora, A. Gasparri, and J. Xiao, "Multirobot tree and graph exploration," *IEEE Transactions on Robotics*, vol. 27, no. 4, pp. 707–717, 2011.

[2] M. Juliá, A. Gil, and O. Reinoso, "A comparison of path planning strategies for autonomous exploration and mapping of unknown environments," *Autonomous Robots*, vol. 33, no. 4, pp. 427–444, 2012.

[3] S. Nuske, S. Choudhury, S. Jain, A. Chambers, L. Yoder, S. Scherer, L. Chamberlain, H. Cover, and S. Singh, "Autonomous exploration and motion planning for an unmanned aerial vehicle navigating rivers," *Journal of Field Robotics*, vol. 32, no. 8, pp. 1141–1162, 2015.

[4] Y. Girdhar, P. Giguere, and G. Dudek, "Autonomous adaptive exploration using realtime online spatiotemporal topic modeling," *The International Journal of Robotics Research*, vol. 33, no. 4, pp. 645–657, 2014.

[5] C. Icking, T. Kamphans, R. Klein, and E. Langetepe, "Exploring simple grid polygons," in *International Computing and Combinatorics Conference*. Springer, 2005, pp. 524–533.

[6] A. Kolenderska, A. Kosowski, M. Małafiejski, and P. Żyliński, "An improved strategy for exploring a grid polygon," in *International Colloquium on Structural Information and Communication Complexity*. Springer, 2009, pp. 222–236.

[7] P. Fraigniaud, L. Gasieniec, D. R. Kowalski, and A. Pelc, "Collective tree exploration," *Networks: An International Journal*, vol. 48, no. 3, pp. 166–177, 2006.

[8] Y. Higashikawa, N. Katoh, S. Langerman, and S.-i. Tanigawa, "Online graph exploration algorithms for cycles and trees by multiple searchers," *Journal of Combinatorial Optimization*, vol. 28, no. 2, pp. 480–495, 2014.

[9] A. Borodin and R. El-Yaniv, *Online computation and competitive analysis*. cambridge university press, 2005.

[10] M. Dunbabin and L. Marques, "Robots for environmental monitoring: Significant advancements and applications," *IEEE Robotics & Automation Magazine*, vol. 19, no. 1, pp. 24–39, 2012.

[11] P. Tokekar, J. Vander Hook, D. Mulla, and V. Isler, "Sensor planning for a symbiotic uav and ugv system for precision agriculture," *IEEE Transactions on Robotics*, vol. 32, no. 6, pp. 1498–1511, 2016.

[12] J. Das, G. Cross, C. Qu, A. Makineni, P. Tokekar, Y. Mulgaonkar, and V. Kumar, "Devices, systems, and methods for automated monitoring enabling precision agriculture," in *Automation Science and Engineering (CASE), 2015 IEEE International Conference on*. IEEE, 2015, pp. 462–469.

[13] P. Tokekar, D. Bhadauria, A. Studenski, and V. Isler, "A robotic system for monitoring carp in minnesota lakes," *Journal of Field Robotics*, vol. 27, no. 6, pp. 779–789, 2010.

[14] P. Tokekar, E. Branson, J. Vander Hook, and V. Isler, "Tracking aquatic invaders: Autonomous robots for monitoring invasive fish," *IEEE Robotics & Automation Magazine*, vol. 20, no. 3, pp. 33–41, 2013.

[15] P. A. Plonski, J. Vander Hook, C. Peng, N. Noori, and V. Isler, "Environment exploration in sensing automation for habitat monitoring," *IEEE Transactions on Automation Science and Engineering*, vol. 14, no. 1, pp. 25–38, 2017.

[16] H. Ishida, Y. Wada, and H. Matsukura, "Chemical sensing in robotic applications: A review," *IEEE Sensors Journal*, vol. 12, no. 11, pp. 3163–3173, 2012.

[17] T. Lochmatter, E. A. Göl, I. Navarro, and A. Martinoli, "A plume tracking algorithm based on crosswind formations," in *Distributed Autonomous Robotic Systems*. Springer, 2013, pp. 91–102.

[18] M. Fahad, Y. Guo, B. Bingham, K. Krasnosky, L. Fitzpatrick, and F. A. Sanabria, "Robotic experiments to evaluate ocean plume characteristics and structure," in *Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on*. IEEE, 2017, pp. 6098–6104.

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

[20] E. M. Arkin, S. P. Fekete, and J. S. Mitchell, "Approximation algorithms for lawn mowing and milling," *Computational Geometry*, vol. 17, no. 1-2, pp. 25–50, 2000.

[21] S. Arya, S.-W. Cheng, and D. M. Mount, "Approximation algorithm for multiple-tool milling," *International Journal of Computational Geometry & Applications*, vol. 11, no. 03, pp. 339–372, 2001.

[22] C. Icking, T. Kamphans, R. Klein, and E. Langetepe, "Exploring an unknown cellular environment." in *EuroCG*, 2000, pp. 140–143.

[23] K. Cesare, R. Skeele, S.-H. Yoo, Y. Zhang, and G. Hollinger, "Multi-uav exploration with limited communication and battery," in *Robotics and Automation (ICRA), 2015 IEEE International Conference on*. IEEE, 2015, pp. 2230–2235.

[24] A. Preshant, K. Yu, and P. Tokekar, "A geometric approach for multi-robot exploration in orthogonal polygons," in *Workshop on Algorithmic Foundations of Robotics (WAFR)*, 2016. [Online]. Available: http://www.wafr.org/papers/WAFR_2016_paper_25.pdf

[25] J. Petrich and K. Subbarao, "On-board wind speed estimation for uavs," in *AIAA Guidance, Navigation, and Control Conference*, 2011, p. 6223.

[26] Z. A. Algfoor, M. S. Sunar, and H. Kolivand, "A comprehensive study on pathfinding techniques for robotics and video games," *International Journal of Computer Games Technology*, vol. 2015, p. 7, 2015.

APPENDIX

A. Proofs of Lemma 2

ALG can be upper bounded as follows:

$$\texttt{ALG} \leq t_{last} + \frac{d_{max}}{S_r - S_p}, \tag{14}$$

where t_{last} is the time that finishes visiting all vertices in the tree and $\frac{d_{max}}{S_r - S_p}$ is the time that traverses the longest length of the backbone when robot and plume moves away from each other. By substituting Equation (1) into the above equation, we have:

$$\leq \frac{2L + (\lfloor \log R \rfloor - 1) d_{max}}{(S_r - S_p)(1 + \lfloor \log R \rfloor)} + \frac{d_{max}}{S_r - S_p}. \tag{15}$$

Using Lemma 1, it becomes:

$$= \frac{2(C - 1 + d_{max} \lfloor \log R \rfloor)}{(S_r - S_p)(1 + \lfloor \log R \rfloor)}. \tag{16}$$

Removing a negative term completes the proof as:

$$\leq \frac{2(C + d_{max} \lfloor \log R \rfloor)}{(S_r - S_p)(1 + \lfloor \log R \rfloor)}. \tag{17}$$

B. Proof of Corollaries

The upper bound for MRSP can simply be obtained by plugging $S_p = 0$ into Equation (2) of Lemma 2.

The upper bound for SRTP can be derived from the upper bound of MRSP by having $R = 0$. However, we can even tighten the bound by using the following observation: if the robot and the plume move toward each other in one direction, they must move away from each other in order to return to the starting location, and vice versa. Therefore, ALG can be upper bounded as:

$$\texttt{ALG} \leq \frac{C - 1}{S_r + S_p} + \frac{C - 1}{S_r - S_p}. \tag{18}$$

Taking out negative terms from the above equation becomes:

$$\leq \frac{2 S_r C}{(S_r + S_p)(S_r - S_p)}, \tag{19}$$

which is a tighter bound than $\frac{2C}{S_r - S_p}$. Note that the difference between these bounds is $\frac{S_r}{S_r + S_p}$ that satisfies $\frac{1}{2} < \frac{S_r}{S_r + S_p} \leq 1$ because $S_r > S_p$.

The upper bound for SRSP can be derived by plugging either $R = 1$ and $S_p = 0$ into the upper bound for MRSP or $S_p = 0$ into the upper bound for SRTP.

C. Proof of Lemma 3

We claim the following inequalities.

$$\texttt{OPT}^R \leq \texttt{OPT}^1, \tag{20}$$

This can be obtained from the fact that the more number of robots are deployed, the shorter time will be taken to explore the entire tree.

Consider a tree consisting of R branches. Then, we claim the following inequality:

$$\texttt{OPT}^1 \leq R\texttt{OPT}^R, \tag{21}$$

Since $\texttt{OPT}^R$ is the time for a robot to explore the longest branch in the tree, $R\texttt{OPT}^R$ must be no less than $\texttt{OPT}^1$.

Combining these inequalities and Equation (3), we prove Lemma 3.