Journal of Intelligent & Robotic Systems (2020) 100:1111–1127
https://doi.org/10.1007/s10846-020-01229-6

Distributed Multi-agent Deployment for Full Visibility of 1.5D and 2.5D Polyhedral Terrains

Aaron Ma[1] · Jorge Cortés[1]

Received: 8 November 2019 / Accepted: 29 June 2020 / Published online: 17 July 2020

Abstract

This paper presents deployment strategies to achieve full visibility of 1.5D and 2.5D polyhedral environments for a team of mobile robots. Agents may only communicate if they are within line-of-sight. In 1.5D polyhedral terrains we achieve this by algorithmically determining a set of locations that the robots can occupy in a distributed fashion. We characterize the time of completion of the resulting algorithm, which is dependent on the number of peaks and the initial condition. In 2.5D polyhedral terrains we achieve full visibility by asynchronously deploying groups of agents who utilize graph coloring and may start from differential initial conditions. We characterize the total number of agents needed for deployment as a function of the environment properties and allow the algorithm to activate additional agents if necessary. We provide lower and upper bounds for the time of completion as a function of the number of vertices in a planar graph representing the environment. We illustrate our results in simulation and an implementation on a multi-agent robotics platform.

Keywords Multi-agent deployment · 2.5 terrain · 1.5 terrain · Visibility · Surveillance · Algorithms · distributed · Planar graphs · Graph coloring · Guarding · Face-spanning subgraph

1 Introduction

With growing interest in autonomous vehicles to aid in complex tasks, including surveillance, disaster response and exploration, this paper considers scenarios where mobile agents are constrained to moving on the ground in 1.5D and 2.5D polyhedral terrains. We consider a fleet of agents who are tasked with guarding these terrains and are constrained by visibility-based communication. Guarding a terrain or region is of particular interest in military applications where mobile or stationary sensors can provide valuable visual feedback regarding activity in the area. The distributed deployment of agents to find guarding locations to collectively maintain complete visibility of a region is well-studied and is known as the 'art gallery' problem. We extend this idea, most commonly studied in 2D environments, to 1.5D and 2.5D terrain environments, where agents determine their respective locations and remain stationary to save battery and focus on surveillance. The autonomous deployment of vehicles to guard environments is becoming an increasingly relevant topic as hardware capabilities and interest in military applications increase. In particular, the emergence of cheap autonomous vehicles make the robust distributed multi-agent deployment approach more attractive.

Consider, for instance, scenarios where agents operate in GPS-denied and unknown environments. In this situation, it is imperative that agents maintain communication connectivity and efficiently explore the environment in a robust manner in case one or more agents fail their mission. It is important that the deployment of vehicles is quick, autonomous, and distributed in order to alleviate user burden. Because of the unknown nature of the environment and the potential presence of adversaries, we are interested RF-quiet missions to avoid the possibility of interception of radio communication. Because of this, we consider line-of-sight communication based on optics. Our objective is to design distributed strategies for deploying robots in polyhedral terrains that achieve

A preliminary version of this paper appeared as [16] at the 2016 ASME Dynamics and Control Conference.

✉ Aaron Ma
aam021@ucsd.edu

Jorge Cortés
cortes@ucsd.edu

1 Department of Mechanical and Aerospace Engineering, University of California, San Diego, CA 92093, USA

complete visibility endowed with explicit guarantees on time completion.

1.0.1 Literature Review

We are inspired by research on distributed algorithms for multi-agent networks [1, 7, 18] addressing computational geometric and optimization problems and, in particular, the classical art-gallery problem [5, 19], which seeks to find the optimum number of guards in a nonconvex environment so that each point is visible to at least one guard. In general, $n/3$ guards are sufficient and sometimes necessary to guard the inside of any polygon with n vertices [9, 24]. The work [11] designs distributed algorithms for teams of robots to guard 2D art gallery environments where agents are constrained to line of sight communication. We extend notions from distributed robotics constrained by line of sight communication as well as work from art-gallery problems to develop deployment for complete visibility in 1.5D and 2.5D environments. In [3], a centralized algorithm is presented for guarding 1.5D terrains in both discrete and continuous domains. The work [15] introduces the terrain guard range as a new geometric parameter by discretizing the definition of range for agents with limited visibility in 1.5D terrains, and proposes a centralized algorithm that solves the 1.5D terrain guarding problem in a tractable fashion with respect to the introduced parameter. In contrast to these methods, we are interested in the distributed deployment of multiple agents to achieve complete visibility in these environments. The work [14] addresses the guarding of polyhedral terrains and proposes methods for calculating and analyzing their visibility. [10] discusses a polynomial-time approximation scheme for guarding of 1.5-dimensional terrains. [13] introduces a centralized, locally optimal, polynomial-time approximation scheme (PTAS) for guarding a terrain. The work [17] explores dynamic and integer linear programming approaches to guard 1.5D and 2.5D terrains also in a centralized setting. Our treatment here builds on ideas from graph coloring and shares commonalities in particular with distributed graph coloring, cf. [4], a problem where each of the nodes on the graph is a static agent with local information consisting of its neighboring nodes. In contrast, in the setting considered here, agents and nodes are two separate entities: agents are deployed starting from a subset of the nodes, eventually all nodes will not be occupied by an agent, and the color of nodes is determined by the agents. The work [25] explores the use of distributed simulated annealing as a technique for 2.5D terrain visibility, where each agent moves to another node with probability based on a temperature function and marginal gain. Similarly, [12] presents a probabilistic algorithm that yields near-optimal results with high probability. In contrast, our proposed algorithm contains rules for agent movement to maintain communication and guarantee complete visibility. The work [6] describes the number of agents required to guard a 2.5D terrain which utilizes a colored planar graph. Recent work [23] employs aerial drones to occupy guarding points in a 2.5D terrain which are determined using graph coloring techniques. In our work here, however, agents are deployed in a distributed fashion under constraints of maintaining visibility-based communication. In order to analyze a 2.5D polyhedral terrain for guarding, we use results from 4-coloring of a planar graph [2]. [22] provides an algorithm for 4-coloring a planar graph which, however, is centralized. Algorithms for 5-coloring a planar graph that are amenable to distributed implementation are found in [8, 26]. A face-spanning subgraph of the 2.5D environment is created as a result of our deployment strategy. Although research in face-spanning subgraphs is sparse, [20] provides results on the minimum number of vertices in a face-spanning subgraph which we use for our results.

1.0.2 Statement of Contributions

We design distributed algorithms for robotic teams to achieve full visibility of polyhedral terrains. Our contributions are structured in two blocks corresponding to 1.5D and 2.5D environments, respectively. The strategies for deployment we propose are iterative processes where the agents communicate through vision, compute, move, and detect where they are in their environment.

For 1.5D environments, we begin by characterizing a guarding set to achieve full visibility of the terrain based on identifying alternate peaks. This allows us to determine a number of agents that are always sufficient and some times necessary to guard any 1.5D environment. Building on this result, we design two deployment strategies and determine closed-form expressions for the time it takes each strategy to finish. The first strategy allows for more flexible initial conditions, while the second strategy completes in less time.

For 2.5D environments, we synthesize a distributed 2.5D deployment strategy that yields complete visibility by utilizing planar graph coloring and redundant locations. We start by identifying locations that are redundant in terms of guarding and visibility. We determine the maximum number of locations that are not redundant (removing an agent that guards a redundant location does not change the collective visibility) and use this result as the sufficient number of agents to guard any 2.5D terrain. Agents are initialized at random nodes in the environment and follow a set of rules to identify, occupy and explore vertices with the objective of achieving complete visibility and connectivity. Agents can be deployed asynchronously at any location and execute the algorithm unaware of other agents other than those which they are connected to via visibility. When one group of connected agents discovers another group, they merge the information collected about the environment so far and will not break communication in the ensuing evolution. Finally, we provide lower and upper bounds on the time completion of the proposed algorithm.

2 Preliminaries

This section introduces basic notation and concepts on planar graphs, coloring, and polyhedral terrains.

2.0.3 Notation

We let $\mathbb{R}$ and $\mathbb{Z}$ denote the set of real and integer numbers, respectively. We denote by $|S|$ the cardinality of the set S. The map ceil : $\mathbb{R} \to \mathbb{Z}$ rounds its argument to the next highest integer. We denote by $\overline{p_1p_2}$ the line segment between points $p_1, p_2 \in \mathbb{R}^d$. A set $\mathcal{C} \subset \mathbb{R}^d$ is convex if the line segment between any pair of its points is contained in $\mathcal{C}$. In $\mathbb{R}^3$, we use x^p, y^p, and z^p to denote the components of the point $p \in \mathbb{R}^3$. Given $p_1, p_2 \in \mathbb{R}^3$, the slope of $\overline{p_1p_2}$ is

$$s_{\overline{p_1p_2}} = \frac{z^{p_2} - z^{p_1}}{\sqrt{(x^{p_2} - x^{p_1})^2 + (y^{p_2} - y^{p_1})^2}}.$$

When convenient, we embed the Euclidean plane $\mathbb{R}^2$ into the Euclidean space $\mathbb{R}^3$ through the map i defined by $i(a_1, a_2) = (a_1, 0, a_2)$. With this embedding, we have $y^p = 0$ for any point $p \in i(\mathbb{R}^2) \equiv \mathbb{R}^2$.

2.0.4 Planar Graphs and Coloring

An undirected graph $\mathcal{G} = (V, E)$ is a pair composed of a vertex set V and an edge set E consisting of bidirectional edges between vertices. The degree of a vertex is the number of edges connected to it. Planar graphs are undirected graphs with vertices in $\mathbb{R}^2$ and whose edges can be drawn on $\mathbb{R}^2$ in such a way that no edges cross each other. A planar graph is *colored* when its vertices are labeled so that no two neighboring vertices share the same label. Planar graphs can be colored with no more than 4 colors, cf. [2]. Centralized algorithms can color planar graphs with no more than 4 colors in $O(n^2)$ time, cf. [22], and with no more than 5 colors in $O(n)$ time, cf. [8]. A face-spanning subgraph is a connected subgraph of $\mathcal{G}$ that contains at least one vertex on every planar face of the graph. A triangulated planar is a planar graph such that the addition of any edge results in a non-planar graph. All bounded faces on a triangulated planar graph are bounded by three edges.

2.0.5 Polyhedral Environments

Polyhedral environments in 1.5D and 2.5D correspond to the graphs of continuous piecewise affine functions on $\mathbb{R}$ and $\mathbb{R}^2$, respectively. Formally, a continuous piecewise affine function $f : I \subset \mathbb{R} \to \mathbb{R}$, with I an interval, defines the 1.5D terrain $S_{1.5}(f) = \{(x, f(x)) : x \in I\} \subset \mathbb{R}^2$. Similarly, a continuous piecewise affine function $f : I \subset \mathbb{R}^2 \to \mathbb{R}$, with I a polygon, defines the 2.5D terrain $S_{2.5}(f) = \{(x, y, f(x, y)) : (x, y) \in I\} \subset \mathbb{R}^3$. When convenient, we drop the dependence on f and simply denote $S_{d.5} \subset \mathbb{R}^{d+1}$, with $d \in \{1, 2\}$, to refer to either of these two cases.

Alternatively, a polyhedral terrain $S_{d.5}$ can be seen as an undirected graph with vertices in $\mathbb{R}^{d+1}$. In the case $d = 1$, these vertices correspond to the points in $\mathbb{R}^2$ where the graph of two affine components of f intersect. In the case $d = 2$, these vertices correspond to the points in $\mathbb{R}^3$ where the graph of three affine components of f intersect. The set of edges connecting vertices in $S_{1.5}$ and $S_{2.5}$ are denoted $E_{1.5}$ and $E_{2.5}$, respectively. For an arbitrary environment, $S_{d.5}$ we specify the respective set of vertices as $V_{S_{d.5}}$. All vertices $v_i \in V_{S_{1.5}}$, except for the extreme ones v_1 and $v_{|V|}$, have degree 2, with neighbors, v_{i-1} and v_{i+1}. It follows that $v \in V_{S_{1.5}}$ are ordered monotonically with respect to the x-axis, such that $x^v_{i-1} < x^v_i < x^v_{i+1}$ for $i \in \{2, \ldots, |V|-1\}$. In $S_{1.5}$, we define $\mathcal{J}_{(v_1,v_2)}$ (resp. $\mathcal{J}_{[v_1,v_2]}$) to be the set of all vertices $v \in V_{S_{1.5}}$ such that $\min(x^{v_1}, x^{v_2}) < x^v < \max(x^{v_1}, x^{v_2})$ (resp. $\min(x^{v_1}, x^{v_2}) \leq x^v \leq \max(x^{v_1}, x^{v_2})$). A vertex v_i in $S_{1.5}$ is a *peak* if $s_{v_{i-1},v_i} > s_{v_i,v_{i+1}}$. Conversely, v_i is a *valley* if it is not a peak. We denote by $\mathcal{P} \subset V_{S_{1.5}}$ and $\mathcal{V} \subset V_{S_{1.5}}$ the collection of peaks and valleys, respectively, in increasing order with respect to their x-coordinate. Given a vertex v we denote its adjacent peak to the right by $p_+(v)$ and to the left by $p_-(v)$.

In our treatment of 2.5D terrains we find it convenient to use triangulated planar graphs. Denote by pr : $\mathbb{R}^3 \to \mathbb{R}^2$ the projection map onto the first two components, $\text{pr}(x, y, z) = (x, y)$. This map projects $S_{2.5}$ onto a planar graph, which we denote by $S^*_{2.5}$. Figure 1 shows a 2.5D terrain $S_{2.5}$ transformed into its planar graph equivalent $S^*_{2.5}$.

Two vertices, v_1 and v_2, are visible to each other if $\overline{v_1v_2}$ does not intersect $S_{d.5}$. We use the following *visibility test* to determine if two vertices v_1, v_2 are visible,

$$s_{\overline{v_1v_2}} > s_{\overline{v_1w}}, \quad \forall w \in K_{v_1v_2}, \tag{1}$$

where, in $S_{1.5}$, $K_{v_1v_2} = \mathcal{J}_{(v_1,v_2)}$ is the set of vertices between v_1 and v_2 and, in $S_{2.5}$, $K_{v_1v_2}$ is the set of points on $S_{2.5}$ that share x and y-coordinates with $\overline{v_1v_2}$. If v_1 is visible to v_2 in $S_{2.5}$, we also refer to the respective vertices in the planar graph $S^*_{2.5}$ as visible. Let $\mathcal{R} \subset V_{S^*_{2.5}}$ such that all vertices in $\mathcal{R}$ are visible to each other. Let $\mathcal{R}_{\text{hull}}$ be the set of vertices that contribute to the convex hull of $\mathcal{R}$ and

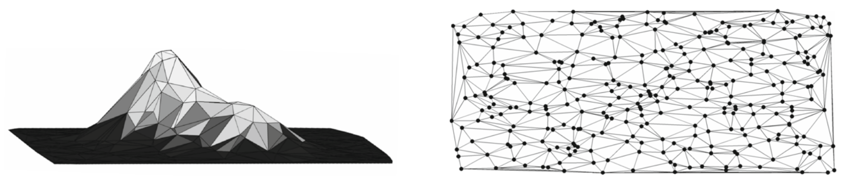


Fig. 1 A 2.5D terrain converted into a planar graph

$\mathcal{R}^* = \mathcal{R} \setminus \mathcal{R}_{\text{hull}}$. We refer to $\mathcal{R}^*$ as a reducible set. We create a new planar graph $S^{**}_{2.5}$ as a modification of $S^*_{2.5}$, where every reducible set in $S^*_{2.5}$ is contracted into a vertex, see Fig. 2 for an illustration. The vertices in V corresponding to those in $S^{**}_{2.5}$ are then $V_{S^{**}_{2.5}}$.

The visibility set of a vertex v in $S^{**}_{2.5}$, denoted $Q(v)$, is the set of all vertices visible to v. Given $V_w \subset V_{S^{**}_{2.5}}$, the collective visibility set,

$$Q(V_w) = \bigcup_{v \in V_w} Q(v),$$

is the set of all vertices visible to them. Finally, $S^{**}_{2.5}$ is fully visible from V_w if $Q(V_w) = V_{S^{**}_{2.5}}$.

3 Problem Statement

We consider scenarios where a team of robots, deployed on an unknown polyhedral terrain $S_{d.5}$, $d \in \{1, 2\}$, seek to achieve full visibility of it. In our treatment, we assume small obstructions such as small rocks and shrubs can be ignored because of their relative minor impact on effective visibility. Instead, we assume that large obstructions such as buildings, large trees, and boulders have already been incorporated in the environment description.

We first describe the model for the robotic network and its capabilities, and then formulate their objective. There are two sets of agents that can be used for deployment. The *active agent set*, A^a, indicates agents that are actively exploring and guarding the environment. The *reserve agent set*, A^r, contains agents that are inactive for purposes of saving energy and resources. Each individual agent has a unique identifier $i \in \{1, \ldots, |A^a| + |A^r|\}$. This provides a sense of priority when two agents decide to execute conflicting actions. The agents are capable of omni-directional vision and can localize vertices at infinite distance. Only agents visible to each other are able to communicate and share information. The agents have the capability to share attributes about vertices such as their local coordinates and color assignments. In $S_{2.5}$, we let agents place relays on a vertex of their choosing to allow communication between any two agents occupying vertices that are neighbors of it. In $S_{1.5}$, agents are able to traverse between two adjacent peaks at every time step. In $S_{2.5}$, agents are able to traverse between vertices connected by an edge at every time step. We consider the motion of the robots slow in comparison to the time required for computation.

We refer to the guarding set, $G \subset V$, as the set of vertices that the agents decide to occupy. This set is determined in a dynamic fashion by the agents as they explore the environment. Our objective is to design coordinated strategies for the robotic team to distributively explore the polyhedral terrain $S_{d.5}$ and determine the guarding set to achieve full visibility. We also seek to characterize both the number of agents and the execution time required by the proposed coordination strategies to achieve this objective.

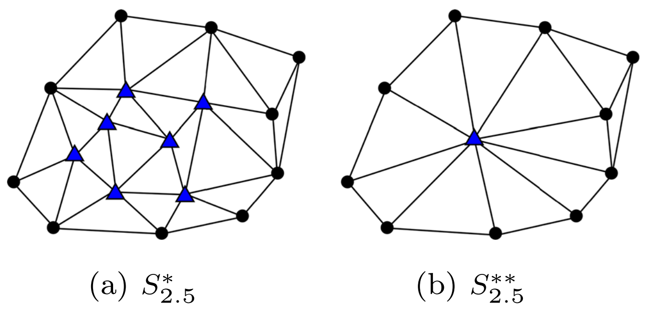


Fig. 2 Contraction of a reducible set in $S^*_{2.5}$. Vertices in $\mathcal{R}^*$ are represented by . $|\mathcal{R}^*| = 7$ is reduced to 1

4 Distributed Deployment over 1.5D Terrains

This section studies the distributed deployment problem over 1.5D polyhedral terrains. We identify a guarding set that guarantees full visibility and study its size. This allows us to obtain a characterization of a sufficient and sometimes necessary number of agents required to complete the task. Building on this characterization, we design strategies to place agents in the identified guarding set.

4.1 Guarding set via Alternate Peaks

We begin our analysis with a simple fact about the visibility regions of adjacent peaks.

Lemma 1 *(Visibility from adjacent peaks): Given two adjacent peaks, v_1 and $v_2 \in \mathcal{P}$, all intermediate vertices of $S_{1.5}$ are visible to them, i.e., $\mathcal{J}_{[v_1,v_2]} \subset Q(v_1) \cap Q(v_2)$.*

Proof Since v_1 and v_2 are adjacent peaks, all vertices $v_i \in \mathcal{J}_{(v_1,v_2)}$ are valleys and have the property, $s_{v_{i-1},v_i} \leq s_{v_i,v_{i+1}}$. Therefore the slope between adjacent vertices, v_i and $v_{i+1} \in \mathcal{J}_{(v_1,v_2)}$, monotonically increases with increasing x along the interval x^{v_1} to x^{v_2}, implying $s_{v_1 v} > s_{v_1 k}$ for all $v \in \mathcal{J}_{(v_1,v_2)}$ and $k \in K_{v_1 v} = \mathcal{J}_{(v_1,v)}$. Hence, all vertices between v_1 and v_2 are visible from either v_1 or v_2. □

As a consequence of Lemma 1, we deduce that $\mathcal{J}_{[p_-(v),p_+(v)]}$ is visible from v. Inspired by this observation, we consider the subset of alternating peaks, denoted $G^{ap} \subset \mathcal{P}$, corresponding to all peaks with odd indices. Note that if $|\mathcal{P}|$ is even, then one can alternatively consider the set of peaks with even indices. The set G^{ap} is the largest set of peaks in $S_{1.5}$ such that every other peak is skipped. This results in $|G^{ap}| = \text{ceil}(|\mathcal{P}|/2)$. We order the indices of G^{ap} in increasing order with respect to their x-coordinate.

Lemma 2 *(Visibility set from alternating peaks): The visibility set of G^{ap} is*

$$Q(G^{ap}) = \begin{cases} V & \text{if } |\mathcal{P}| \text{ odd}, \\ \mathcal{J}_{[v_1,p_{|\mathcal{P}|}]} & \text{if } |\mathcal{P}| \text{ even}. \end{cases}$$

Proof From Lemma 1, if v_1 and v_2 are two peaks with a single peak v between them (e.g., $p_+(v_1) = v = p_-(v_2)$), then $Q(v_1 \cup v_2)$ contains $\mathcal{J}_{[p_-(v_1),p_+(v_2)]}$. It follows that, if $G^{ap} = \{g_1, \ldots, g_{|G^{ap}|}\}$, then $Q(G^{ap})$ is equal to $\mathcal{J}_{[p_-(g_1),p_+(g_{|G^{ap}|})]}$, and the result follows. □

As a consequence of this result, we identify $G^{ap} \cup \{p_{|\mathcal{P}|}\}$ as a sufficient set of vertices to achieve full visibility.

Theorem 1 *(Full visibility): The 1.5D environment $S_{1.5}$ is fully visible from $G = G^{ap} \cup \{p_{|\mathcal{P}|}\}$. Furthermore, $|G| = \text{floor}(|\mathcal{P}|/2) + 1$ is sufficient and sometimes necessary to achieve full visibility of $S_{1.5}$.*

Proof The fact that $Q(G) = V$ readily follows from Lemma 2. If $|\mathcal{P}|$ is odd, then $G = G^{ap}$ and therefore $|G| = \text{ceil}(|\mathcal{P}|/2) = \text{floor}((|\mathcal{P}|/2) + 1$. If $|\mathcal{P}|$ is even, $|G| = \text{ceil}(|\mathcal{P}|/2) + 1 = \text{floor}((|\mathcal{P}|/2) + 1$. To show that $|G|$ agents are sometimes necessary, we provide a specific example. Consider an environment where all vertices are peaks. Then, the visibility set of any vertex v is exactly $Q(v) = \mathcal{J}_{[p_-(v),p_+(v)]}$, which implies that any guarding set must contain at least every other vertex in order to achieve full visibility. □

We specify environments with ratio of peaks to valleys to be $1 : 1$, as an average case scenario. In what follows, we allocate resources for this average case scenario, but allow for flexibility if needed.

4.2 1.5D alternate peak strategy

Given our analysis in Section 4.1, here we design distributed strategies to deploy agents on $G = G^{ap} \cup \{p_{|\mathcal{P}|}\}$. We initially begin with $|A^a| = \text{floor}(|V_{S_{1.5}}|/4) + 1$ as this is sufficient for average case scenarios, and $|A^r| = \text{floor}(|\mathcal{P}|/2) + 1 - |A^a|$ in case the ratio of peaks to valleys is worse than expected. We begin with an informal algorithm description.

[*Informal description*]: All agents are initially located at vertex, v_0, whose position in $S_{1.5}$ is unknown to them. Agents explore $S_{1.5}$ and incrementally distribute themselves on $G = G^{ap} \cup \{p_{|\mathcal{P}|}\}$. Half of the agents, $A_{\text{lft}} \in A^a$, go left, while the other half, $A_{\text{rght}} \in A^a$, goes right (if $|A|$ is odd, we let A_{lft} have one extra agent). Depending on the location of v_0 within the environment, one of these two sets contains too many agents. Agents keep track of a variable termed "goal". Once an agent detects the edge of $S_{1.5}$ (either v_1 or $v_{|V|}$), it raises its "goal" flag, which signals visible neighboring agents that the other group needs more agents to complete the algorithm. Two strategies are then possible. Let A_- be the group of agents that does not have enough agents, and A_+ be the group that has too many. In both strategies, A_- deploys until they guard as many alternating peaks as they can. Then, in the ***1.5D alternate peak strategy with wait***, agents in A_- wait until they receive a "goal" message from A_+ to continue exploring and finally guarding $S_{1.5}$. Instead, in the ***1.5D alternate peak strategy w/o wait***, agents in A_- make the assumption that the "goal" flag will eventually come from A_+ and continue deploying towards the boundary of $S_{1.5}$ (creating a void in visibility coverage that will eventually be filled by the agents in A_+). If at any time v_0 runs low on active agents without the task having been completed, this means that the ratio of peaks to valleys in the environment is higher than expected and 2 more agents are activated from A^r to provide resources to the left and right groups.

Algorithm 1 provides a formal description of ***1.5D alternate peak strategy***, both `with` and `without wait`. The steps that are only executed under ***1.5D alternate peak strategy w/o wait***are marked with the symbol †. All other steps are common to both strategies.

Remark 1 (Wait versus no wait): The strategies differ in how the agents react when they determine that there are not enough agents in their group to reach the boundary of the environment. While the ***1.5D alternate peak strategy w/o wait***completes in less time, it requires all agents to start on the same initial condition (otherwise the use of the "continue" flag might be detrimental to algorithm

Algorithm 1 ***1.5D alternate peak strategy*.**

```
Agent $a$ variables:
bool $goal$=False, $continue$=False
int $direction$=-1 | $a \in A_{\text{lft}}$ or 1 | $a \in A_{\text{rght}}$
 While $Q(G)$ is not $V$:
Communicate
      if any visible agents to $a$ have $goal$ is True:
          $a$ sets $goal$ to True
          $a$ sets $direction$ to $direction$ of agent with $goal$ to True
Move
      if any of the following conditions are met:
      • Agent $a$ occupies $v \notin \mathcal{P}$
      • $a \in A_{\text{lft}}$ and $\mathcal{J}_{(v,p_+(v)]}$ is occupied or $a \in A_{\text{rght}}$ and $\mathcal{J}_{[p_-(v),v)}$ is occupied
      • $a$ does not have the greatest ID of all agents that occupy $v$
      • †: $a$ has $goal$ is False and $continue$ is True
              if $a$ is only active agent that occupies $v$ and inactive agents occupy $v$:
      activate two agents from $A^r$
          $a$ moves one peak dictated by $direction$
      else:
          $a$ stays at vertex $v$
Detect
      if $v_1$ or $v_{|V|}$ is visible:
          $a$ sets $goal$ to True
          $a$ sets $direction$ away from detected $v_1$ or $v_{|V|}$
†: if the time elapsed is equal to $2|A_-|+a$.ID$-2$
      $a$ sets $continue$ to True
```

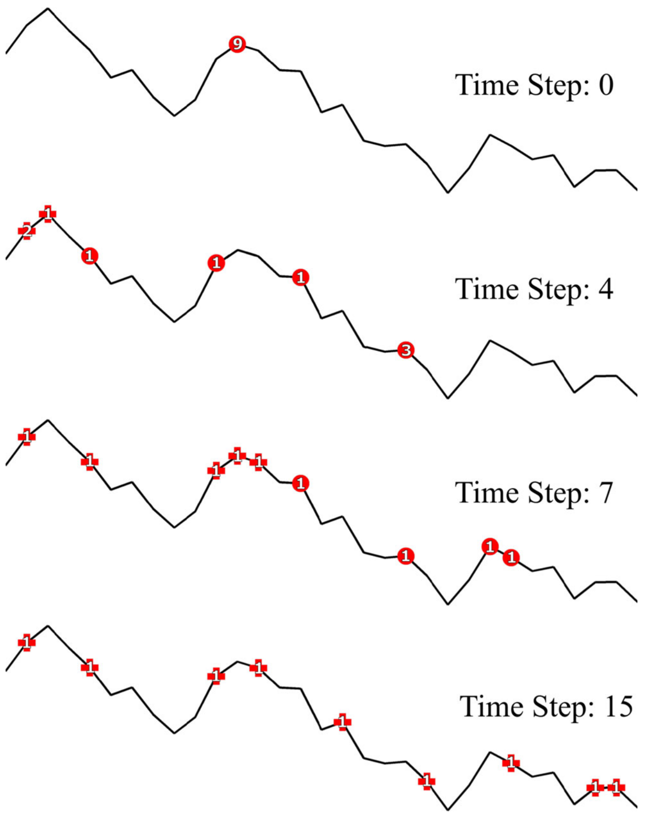


Fig. 3 Execution of ***1.5D alternate peak strategy with wait***on a 1.5D environment with 16 peaks. From Theorem 1, $|A| = 9$ agents are sufficient to achieve full visibility. All agents begin at the same initial location, v_0, of index 6 with respect to $\mathcal{P}$, and split into two groups. The location of agents with the "goal" flag not raised are shown by a red dot where the number states the number of agents on that vertex. Agents with a raised "goal" flag are denoted with a cross

completion). Instead, the ***1.5D alternate peak strategy with wait***requires in general more time to complete, but agents can be initialized at multiple locations.

Remark 2 (Ordering of agents): Agents occupy a peak only if no other agent with lower ID occupies the same peak. As the algorithm executes, the agents naturally order themselves within their respective groups of A_- and A_+ in decreasing order of ID from v_0 in the direction they are initialized. This enables the agents to rationalize when the "goal" flag should have arrived by (agents in A_+ with lower ID receive the "goal" flag before agents with greater ID). Due to the speed at which the "goal" flag propagates in A_+, agents in A_- rationalize that they are not in A_+ if they do not receive the "goal" flag in $2A_- +$ ID -2 time steps.

Figure 3 shows an example of agents being deployed on $S_{1.5}$ using the ***1.5D alternate peak strategy with wait***. At time step: 4, A_+ reaches the leftmost boundary and raises the "goal" flag. At time step: 7, A_- runs out of agents and begins to wait for the "goal" flag. By time step: 15, the network has completely deployed achieving full visibility of the environment.

4.3 Time Steps for Algorithm Completion

In this section we characterize the number of time steps required by the proposed strategies for completion. We recall that an agent can move between adjacent peaks in one time step. For the following analysis, let i be the index of v_0 in $\mathcal{P}$ and define

$$i^* = \begin{cases} i & \text{if } i \leq |\mathcal{P}|/2, \\ |\mathcal{P}| - i + 1 & \text{if } i > |\mathcal{P}|/2. \end{cases} \tag{2}$$

The following three sets cover all possibilities for the location of the initial vertex v_0,

$$\begin{aligned} \mathcal{A} = \{v_0| & \text{ if } |A| \text{ even, } i \leq |\mathcal{P}|/6 + 1 \text{ or } i \geq 5|\mathcal{P}|/6 - 1 \\ & \text{ and if } |A| \text{ odd, } i \leq |\mathcal{P}|/6 \text{ or } i \geq 5|\mathcal{P}|/6\}, \\ \mathcal{B} = \{v_0| & \text{ if } |A| \text{ even, } |\mathcal{P}|/6 + 1 < i < 5|\mathcal{P}|/6 - 1 \\ & \text{ and if } |A| \text{ odd, } |\mathcal{P}|/6 < i < 5|\mathcal{P}|/6\}, \end{aligned}$$

and region $\mathcal{C}$, which only exists if $|\mathcal{P}|$ is odd and corresponds to $i = \frac{|\mathcal{P}|+1}{2}$. Figure 4 illustrates these three cases.

We are ready to characterize the time complexity of the strategy with wait.

Theorem 2 *(**1.5D alternate peak strategy with wait**completion time): The number of time steps required by the **1.5D alternate peak strategy with wait**to complete is*

$$T = \begin{cases} |\mathcal{P}| - i^* & v_0 \in \mathcal{A}, \\ |\mathcal{P}| + \frac{i^*}{2} - |A_-| - \frac{3}{2} & v_0 \in \mathcal{B}, \\ \frac{3(|\mathcal{P}|-1)}{4} & v_0 \in \mathcal{C}, \end{cases} \tag{3}$$

where i^* is determined by v_0 according to Eq. 2.

Proof For simplicity of exposition, we only consider the case when $i \leq (|\mathcal{P}|+1)/2$ (the case when $i > \frac{|\mathcal{P}|}{2}$ is analogous). Consequently, $i^* = i$, $A_{\text{lft}} = A_+$, and $A_{\text{rght}} = A_-$. We first consider the scenario when both groups of agents reach the boundary of the environment at the same time. Note that this is only possible if $v_0 \in \mathcal{C}$.

Case $\mathcal{C}$: If $|\mathcal{P}|$ is odd and v_0 is the peak with index $\frac{|\mathcal{P}|+1}{2}$, the agents split up perfectly since there are the same number of peaks to the left and right. The agents reach the boundaries and send the "goal" message at the same time. The algorithm completes when the goal messages meet at $\frac{|\mathcal{P}|+1}{2}$. The time to completion is then the sum of the time to reach the boundaries, and the time that it takes for the flags to reach the $\frac{|\mathcal{P}|+1}{2}$, which is

$$\frac{|\mathcal{P}|+1}{2} - 1 + \frac{\frac{|\mathcal{P}|+1}{2} - 1}{2} = \frac{3(|\mathcal{P}|-1)}{4}.$$

Next, we consider the scenario when both groups of agents do not reach the boundary of the environment at the same time. In this scenario, it is A_+ which reaches the boundary first. Agents move one peak at a time, distributing themselves on every other peak. Because of this, note that A_- runs out of agents after exactly $2|A_-|$ time steps. On the other hand, it takes exactly $i^* - 1$ time steps for agents in A_+ to reach the boundary of $S_{1.5}$ and raise the "goal" flag. At this time, the rightmost agents in A_- are located at peak

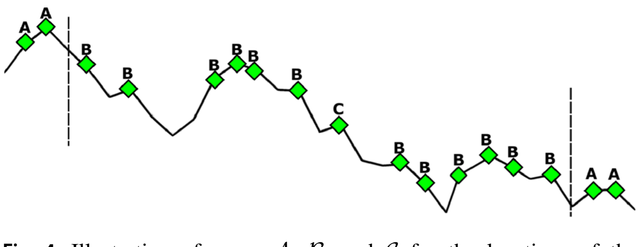


Fig. 4 Illustration of cases $\mathcal{A}$, $\mathcal{B}$, and $\mathcal{C}$ for the locations of the common initial condition of the agents. The 1.5D environment $S_{1.5}$ has $|\mathcal{P}| = 17$ peaks.

$i^* + (i^* - 1) = 2i^* - 1$. Once the "goal" flag is raised, since agents can communicate with agents at adjacent peaks, the speed at which the "goal" flag is communicated is effectively two peaks per time step.

Two things might happen depending on whether the "goal" flag reaches the rightmost agents in A_- before this group runs out of agents. Let t denote the number of time steps elapsed since A_+ first raised the "goal" flag. After t time steps, the goal flag is at $1 + 2t$. If A_- does not run of agents, its rightmost agents are at $2i^* - 1 + t$. Therefore, we are looking for the solution to $1 + 2t = 2i^* - 1 + t$, which is

$$t = 2i^* - 2.$$

The total elapsed time since the beginning is then $i^* - 1 + (2i^* - 2) = 3i^* - 3$. This time must be less than or equal to than the time it takes A_- to run out of agents, i.e.,

$$i^* \leq \frac{2}{3}|A_-| + 1. \tag{4}$$

Case $\mathcal{A}$: One can see that Eq. 4 is satisfied if and only if $v_0 \in \mathcal{A}$. Because the "goal" flag reaches the rightmost agent in A_- before A_- runs out of agents, A_- moves at one time step towards the rightmost boundary through the entirety of the strategy. Once A_- reaches the boundary, the agents will have distributed themselves on G. Therefore, if $v_0 \in \mathcal{A}$, we deduce that the number of time steps required for completion is $T = |\mathcal{P}| - i^*$.

Case $\mathcal{B}$: If instead, $v_0 \in \mathcal{B}$, this means that Eq. 4 is not satisfied, i.e., A_- runs out of agents before the "goal" flag reaches its rightmost agents. After the "goal" flag is raised, agents in A_+ move at 1 peak per time step and occupy their half of G by the time the "goal" flag reaches the rightmost boundary since no agent has to travel more than $|\mathcal{P}|/2$ peaks. Since agents in A_- previously occupy alternating peaks, they all must travel the same number of peaks to reach their final configuration. The rightmost agent in A_- receives the "goal" flag last and is the last agent to occupy its peak in G. Therefore, we need to compute the time it takes for A_- to receive the message and the leftover time needed for the rightmost agent in A_- to move to the boundary of $S_{1.5}$. A_- runs out of agents at vertex $d = i^* + 2|A_-|$. With the notation used above, the time required for the "goal" flag to reach this vertex is the solution to $1 + 2t = d$, i.e., $t = (d-1)/2$. Once the A_- has received the message it takes

$$|\mathcal{P}| - d,$$

steps to reach the boundary. Therefore, the total number of time steps is

$$T = i^* - 1 + (d-1)/2 + |\mathcal{P}| - d = |\mathcal{P}| + \frac{i^*}{2} - |A_-| - \frac{3}{2}.$$

□

From Theorem 2, one can see that, in region $\mathcal{B}$, the time complexity monotonically increases as the initial location moves from the left boundary of this region (at $|\mathcal{P}|/6+1$ or $|\mathcal{P}|/6$, depending on whether $|A|$ is even or not), to the peak closest to $\frac{|\mathcal{P}|}{2}$, Next, we determine the completion time of the ***1.5D alternate peak strategy w/o wait***.

Theorem 3 *(**1.5D alternate peak strategy w/o wait**completion time): The number of time steps required for the **1.5D alternate peak strategy w/o wait**to complete is*

$$T = \begin{cases} |\mathcal{P}| - i^* & v_0 \in \mathcal{A}, \\ \frac{7|\mathcal{P}|}{8} - \frac{i^*}{4} & v_0 \in \mathcal{B}, \\ \frac{3(|\mathcal{P}|-1)}{4} & v_0 \in \mathcal{C}, \end{cases} \tag{5}$$

where i^* is determined by v_0 according to Eq. 2.

Proof With respect to the proof of Theorem 2, the scenarios when the initial condition belongs to $\mathcal{A}$ and $\mathcal{C}$ are the same. In case $\mathcal{A}$, the "goal" flag is communicated completely before either group runs out of agents. Similarly, in case $\mathcal{C}$ the algorithm completes before the "continue" flag is raised.

Case $\mathcal{B}$: In this case, agents in A_- rationalize as soon as possible that they are in A_- and raise their "continue" flag. Agent, a, in A_- raises its "continue" flag at time step: $2A_-+$ID -2, where ID is the unique identification of the agent. Since the agents occupy peaks in order of decreasing ID, this is the amount of time the agent should have received the "goal" flag if it belonged in A_+. a then moves at one peak per turn in its initial direction until it receives or raises the "goal" flag. This means that if v_0 is in region $\mathcal{B}$, two "goal" flags may be active at the same time and propagate from the boundaries to the center. The algorithm terminates when agents no longer move, which occurs where the two "goal" flags meet. We use this fact to determine the time of execution for ***1.5D alternate-peak strategy w/o wait***in region $\mathcal{B}$. For this analysis, we examine the "goal" flag from A_-. The rightmost agent in A_- reaches the boundary and raise the "goal" flag in $t_1 = |P| - i^*$ since the ***1.5D alternate peak strategy w/o wait***allows agents to move at one peak per time step to the boundary when "continue" is raised. The rightmost "goal" flag travels at two peaks per time step and meets the leftmost "goal" flag in $t_2 = \frac{d}{2}$ time steps, where d is the distance from the rightmost boundary to where the "goal" flags meet. Note that d increases linearly with respect to i^* since the time that $|A_-|$ and $|A_+|$ reach their respective boundaries is linearly dependent on i^*. We interpolate between two boundary initial conditions of region $\mathcal{B}$ to determine d. For $v_0 = \frac{|P|}{2}$, A_- and A_+ reach their respective boundaries at or around the same time. The goal flags meet at the center of $S_{1.5}$, and $d = \frac{|P|}{2}$. For $v_0 = \frac{|P|}{6}$, the goal flag from A_+ reaches the rightmost agent in A_- just as the "continue" flag is raised and meets with rightmost "goal" flag at $d = 0$. We interpolate for:

$$d = \frac{3}{4}(i^* - \frac{|\mathcal{P}|}{6}),$$

which allows us to determine the time of execution

$$T = t_1 + t_2 = |\mathcal{P}| - i^* + \frac{3}{4}(i^* - \frac{|\mathcal{P}|}{6}) = \frac{7|\mathcal{P}|}{8} + \frac{i^*}{4},$$

as stated.

□

We find that ***1.5D alternate peak strategy w/o wait*** completes quicker than ***1.5D alternate peak strategy with wait*** in regions $\mathcal{B}$ and $\mathcal{C}$. In region $\mathcal{A}$, the limiting factor for both algorithms is the time required to traverse $S_{1.5}$ from one end point to the other. These realizations are illustrated in Fig. 5 which graphs results from Theorems 2 and 3 with respect to the initial starting vertex.

5 Distributed Deployment over 2.5D Terrains

This section studies the distributed deployment problem over 2.5D polyhedral terrains. We introduce the concept of a (non-)redundant vertex of a guarding set and characterize a sufficient and sometimes necessary number of vertices of guarding sets without redundant vertices. We build on this result to design a distributed strategy to efficiently

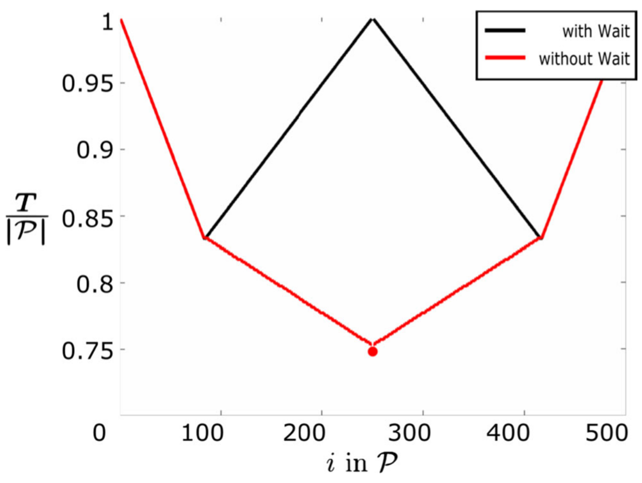


Fig. 5 Time of execution versus initial condition in a 1.5D environment with 500 peaks. On the y-axis we show the time of execution T over the number of peaks $|\mathcal{P}|$. The x-axis describes the initial starting location v_0

place the robotic agents and achieve full visibility as shown in Fig. 10.

5.1 Guarding set via Non-redundant Vertices

In this section, we reason over the planar graph $S^{**}_{2.5}$ determined by contracting reducible sets in $S^{*}_{2.5}$ as described in Section 2.0.4. Let $\Delta_{S^{**}_{2.5}}$ be the set of triangles in the triangulated planar graph $S^{**}_{2.5}$ and let G be a guarding set. A triangle Δ is uniquely guarded if only 1 of its vertices v is in G. We denote the collection of such triangles by $\Delta_u \subset \Delta_{S^{**}_{2.5}}$ and by Δ^v_u the set of all triangles uniquely guarded by v. A vertex in $V_{S^{**}_{2.5}}$ is *redundant* if it does not uniquely guard any triangle, i.e., $|\Delta^v_u| = 0$. An edge in $S^{**}_{2.5}$ is *redundant* if it connects two redundant vertices. A vertex is *non-redundant* if it is not redundant, i.e., $|\Delta^v_u| \geq 1$. We denote by V^r and $E_r \subset E_{S^{**}_{2.5}}$ the set of redundant vertices and edges, respectively. Let G_{nr} be the result of removing all redundant vertices from G. Note that G_{nr} remains a guarding set and, furthermore, $G_{nr} = V_{S^{**}_{2.5}} \setminus V^r$.

Lemma 3 (Upper bound on $|E_r|$) *Given a guarding set G of $S^{**}_{2.5}$, $|E_r| \leq 3|V^r| - 6$.*

Proof For planar graphs, the maximum number of edges given n vertices is $3|n|-6$. Since $E_r \subset E_{S^{**}_{2.5}}$, no edges cross in E_r, thus the graph (V^r, E_r) is also planar. □

In order to bound the number of agent needed for achieving full visibility on $S^{**}_{2.5}$, we next determine the minimum number $|E_r|$ of redundant edges that can exist given G. To do this we find the maximum ratio of non-redundant vertices to redundant edges. Given $e_r \in E_r$, let $N_{e_r,\Delta}$ be the number of uniquely guarded triangles that are adjacent to e_r. Note that, for any planar graph, we have $N_{e_r,\Delta} \leq 2$. This is because for any redundant edge, there exists at most two vertices v_1 and v_2 in G that uniquely guard triangles adjacent to e_r. Let

$$r_{e_r} = \begin{cases} \frac{1}{\Delta^{v_1}_u} & \text{if } N_{e_r,\Delta} = 1, \\ \frac{1}{\Delta^{v_1}_u} + \frac{1}{\Delta^{v_2}_u} & \text{if } N_{e_r,\Delta} = 2. \end{cases} \tag{6}$$

Since the minimum number of triangles that any vertex $v \in G_{nr}$ uniquely guards is 1, $|\Delta^v_u| \geq 1$ and, therefore, $r_{e_r} \leq 2$. Furthermore, the number of vertices in G_{nr} can be expressed as the sum of this ratio for all redundant edges

$$|G_{nr}| = \sum_{e_r \in E_r} r_{e_r}. \tag{7}$$

Using $r_{e_r} \leq 2$, this quantity is bounded as

$$|G_{nr}| \leq 2|E_r|. \tag{8}$$

Theorem 4 (Upper bound on $|G_{nr}|$) *Let G_{nr} be a guarding set of $S^{**}_{2.5}$ without redundant vertices. Then $|G_{nr}| \leq (|6V_{S^{**}_{2.5}}| - 12)/7$.*

Proof Using Lemma 3 and Eq. 8, $|G_{nr}| \leq 2|E_r| \leq 6|V^r| - 12$. From $G_{nr} = V_{S^{**}_{2.5}} \setminus V^r$, we have $|V^r| = |V_{S^{**}_{2.5}}| - |G_{nr}|$. Then

$$|G_{nr}| \leq 6|V_{S^{**}_{2.5}}| - 6|G_{nr}| - 12$$

and the result follows. □

The bound in Theorem 4 can be improved with an additional operation after pruning redundant vertices. Consider the particular case where there are two non-redundant vertices v_1 and v_2 that uniquely guard triangles adjacent to a single redundant edge e_r and that $|\Delta^{v_1}_u|$ and $|\Delta^{v_2}_u|$ are 1. In this case, we call e_r a *doubly redundant edge*, as is the type of edge that generates the maximum possible ratio $r_{e_r} = 2$. When there is a doubly redundant edge, it is always possible to convert both v_1 and v_2 to redundant vertices and convert one of the original redundant vertices to non-redundant. Doing this operation does not change the overall guarding character of the modified G_{nr}, since both of the uniquely guarded triangles are still guarded by the new non-redundant vertex. We call this process ***Non-redundant vertex reduction***, which can be executed on $S^{**}_{2.5}$ sequentially for each doubly redundant edge. Let the remaining guarding set after removing doubly redundant edges via ***Non-redundant vertex reduction***be G^*_{nr}.

Theorem 5 *(Upper bound on $|G^*_{nr}|$): Let G^*_{nr} be a guarding set of $S^{**}_{2.5}$ determined by running **Non-redundant vertex reduction**on G_{nr}. Then $|G^*_{nr}| \leq (|9V_{S^{**}_{2.5}}| - 9)/11$.*

Proof This proof follows that of Theorem 4 except that the ratio between non-redundant vertices and redundant edges changes. After ***Non-redundant vertex reduction***is executed, no doubly redundant vertices exist and r_{e_r} is maximized when $N_{e_r,\Delta} = 2$ and one of its associated non-redundant vertices v uniquely guards at least 2 or more triangles, $|\Delta^v_u| \geq 2$. In this case, the maximum possible ratio according to Eq. 6 is $r_{e_r} = 1.5$. Using this fact in Eq. 7 yields

$$|G^*_{nr}| \leq 1.5|E_r|.$$

A similar reasoning as in the proof of Theorem 4 yields the desired result. □

The bound in Theorem can be further refined for a class of environments that satisfies a specific topological

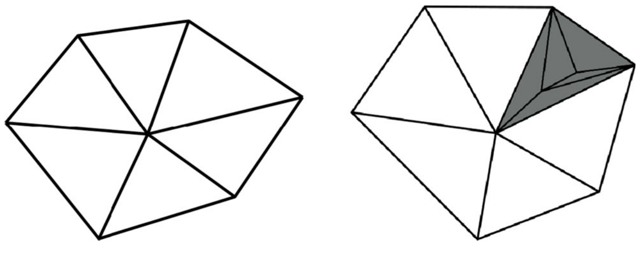


Fig. 6 A simple illustration of two $S_{2.5}^{**}$ graphs with and without embedded triangles. In **a** $N^r = 0$ and in **b** $N^r = 1$

assumption. We say $S_{2.5}^{**}$ contains *embedded triangles* if there exist a triangle that is contained in the convex hull of another triangle in $S_{2.5}^{**}$, cf. Figure 6. We denote by N^r the number of triangles in $S_{2.5}^{**}$ that encapsulate 1 or more triangles in its convex hull.

Theorem 6 *(Upper bound on $|G_{nr}^*|$ with* ***Non-redundant vertex reduction****over environments without embedded triangles): Let G_{nr}^* be a guarding set of $S_{2.5}^{**}$ determined by running* ***Non-redundant vertex reduction****on G_{nr} and assume that $S_{2.5}^{**}$ does not contain embedded triangles. Then $|G_{nr}^*| \leq (3|V_{S_{2.5}^{**}}| - 4)/4$.*

Proof To guard any given triangle, one of its vertices must belong to G_{nr}^* if $S_{2.5}^{**}$ contains no embedded vertices (this is not necessarily true when $S_{2.5}^{**}$ does contain embedded triangles) This implies that there cannot be any redundant vertices that form a triangle. Because the resulting graph of redundant edges is planar, and planar graphs without triangles have no more than $2n - 3$ edges (with n being the number of vertices), then $|E_r| \leq 2|V_{S_{2.5}^{**}}| - 3$. Using this inequality instead of Lemma 3 in the proof of Theorem yields the result. □

We rely on the results above to upper bound the number of agents that need to be active in the guarding deployment of $S_{2.5}^{**}$.

5.2 $S_{2.5}$ Exploration and Guarding Algorithm

If we know that there are no embedded triangles in the environment, then we let $|A^a| = (3|V_{S_{2.5}^{**}}| - 4)/4$ and $|A^r| = \emptyset$ based on Theorem 6. Otherwise, if we have no knowledge regarding the presence of embedded triangles in the environment, we consider initially $|A^a| = (3|V_{S_{2.5}^{**}}| - 4)/4$ and $|A^r| = (9|V_{S_{2.5}^{**}}| - 9)/11 - |A^a|$ agents based on Theorem . Our proposed deployment strategy seeks to guard non-redundant vertices to achieve distributed coverage of $S_{2.5}$. The idea is to allocate agents on a guarding set G_{nr}^* determined by a valid coloring of $S_{2.5}^{**}$ and detection of redundant vertices. To determine a valid coloring of $S_{2.5}^{**}$, the agents label vertices that they discover such that no connected vertices are labeled the same color. We refer to the colors that the agents assign to vertices as $\{1, 2, G_c\}$, where G_c is a label for a color in $\{3, 4, 5, 6\}$ that is instantiated during the execution of the algorithm and denotes vertices that agents plan to occupy or place a relay on. By definition of coloring on planar graphs, each triangular face in $S_{2.5}^{**}$ contains a vertex that is colored G_c. Occupying all vertices that are not $\{1, 2\}$ guarantees full visibility, since every face on $S_{2.5}$ is a triangle.

Agents arrange themselves into trees to organize the exploration in a hierarchical fashion. We use the following notation to indicate properties of an agent including their determined place in the dynamically created trees

$a.v$: The vertex v occupied by a
a.parent: An agent that is the parent of a
a.children: A set of agents that are children of a
a.root: True if a is the root of a tree
a.explored: True if a recently explored

In addition to knowledge regarding their tree, agents need to keep track of which vertices have been explored thus far. We sort vertices in $S_{2.5}$ into *exploration* sets, which help the agents decide which vertices to explore. Let us introduce some terminology and notation. Vertices that are have not yet been detected by agents belong to the *undetected* set U. If v has been detected by the agents and all of the triangles that touch v are visible to agents, then v belongs to the *known* set of vertices K. Finally if a vertex v has been detected but some triangles that touch v are not visible, i.e., $v \notin$ K, then v belongs to the *discovered* set D. All agents connected via edges on $S_{2.5}$ are able to communicate. Note that, at any given time, there may be more than one connected component of agents. Each connected component A_i^C maintains its own exploration sets of vertices, $\mathrm{U}_{A_i^C}$, $\mathrm{D}_{A_i^C}$, and $\mathrm{K}_{A_i^C}$.

We proceed to define ***2.5D non-redundant peak strategy***. The following is an informal description for the pseudocode in Algorithm 2.

[*Informal description*]: Agents can be initialized on any vertex. Connected components of agents are determined based on connectivity of the initial condition. Each of the occupied vertices are assigned color G_c. There are multiple processes for each step of ***2.5D non-redundant peak strategy*** , which each connected component of agents execute. The agents first create *subtrees* in ***build_subtrees***, that allow them to organize themselves for exploration and maintaining connectivity. The connected component of agents A_i^C keeps track of $D_{A_i^C}$ and $K_{A_i^C}$. Agents color newly discovered vertices in ***color***. In ***explore***, one agent in each of the created subtrees move to a vertex in $D_{A_i^C}$. The vacant spot is then occupied by the parent of the agent that moves. Agents then use ***redundant_agent_removal***to refine the guarding set so that $|A|$ is sufficient. We allow the agents to place relays, $r \in R$ at a vertex they occupy. Relays provide communication between neighboring vertices. ***2.5D non-redundant peak strategy***repeats until agents occupy G such that $Q(G) = V$. As a result of the exploration, two or more separate connected components of agents may be able to communicate with each other via neighboring edges. In ***merge***, their exploration sets, $K_{A_{C,i}}$, $U_{A_{C,i}}$ and $D_{A_{C_i}}$ are combined and the sets of connected components of agents merge into one, larger set. Last, ***agent_activation***activates agents from the set of reserve agents when embedded triangles are detected in $S^{**}_{2.5}$.

Algorithm 2 ***2.5D non-redundant peak strategy***.

```
for all $a$:
  $a.parent = None$
  $a.children = \emptyset$
  $a.v$ = vertex that agent occupies
def main:
  While $Q(G)$ is not $V$:
      build_subtrees
      color
      explore
      redundant_agent_removal
      merge
      agent_activation
```

5.2.1 Process: build_subtrees

The first process of ***2.5D non-redundant peak strategy***is **build_subtrees** where agents structure themselves for exploration.

[*Informal description*]: In ***build_subtrees***agents determine tree structures that allow them to explore without breaking connectivity. This algorithm begins by determining which vertices are occupied by more than one agent. One agent on these vertices claim to be the *root* of a new *subtree*. Agents can only belong to one subtree. After roots determined, the agents take turns adopting neighboring agents as their children. Each agent maintains memory of their parent and child. If an agent has no children they will have priority in exploring amongst its subtree in ***explore***.

build_subtrees.

```
find_roots ()
While $\mathcal{AP}$ is not empty:
    $a = \mathcal{AP}.get()$
    adopt_children($a$)
    Subfunctions
def find_roots
    $a.parent$ = None
    $a.children = \emptyset$
    $\mathcal{AP} = \emptyset$
    If $a.v$ is occupied by at least one other agent and no
other agents are a root:
      $a.root = True$
      $\mathcal{AP} = \mathcal{AP} \cup a$
def adopt_children(a):
    for all $b$ that neighbors $a$:
      if $b.parent$ =None:
        $a.children$.append($b$)
        $b.parent = a$
        $\mathcal{AP}.put(b)$
```

5.2.2 Process: color

Next, we describe the process for coloring vertices that have just been discovered.

[*Informal description*]: Agents that have recently explored (a.explored = $True$) color their neighboring vertices in this process. For each agent a, if a neighboring vertex v is in $D_{A_i^C}$ and that vertex has no neighbors with color *1*, then a colors that vertex *1*. The agent does the same for color *2*. If v has neighbors of both colors *1* and *2*, then a colors that vertex G_c, indicating that vertex should be guarded in future exploration steps.

color

```
for $a \in A$ such that $a$.explored= $True$:
    for all $v \in D_{A_i^C}$ that neighbor $a$:
      if $v$ has no neighbors with color 1:
        $a$ colors $v$ to 1
      else if $v$ has no neighbors with color 2:
        $a$ colors $v$ to 2
    else:
        $a$ colors $v$ to $G_c$
    $a$.explored= $False$
    $a$ broadcasts and updates colors in $D_{A_i^C}$
    $a$ sets $D_{A_i^C}$ to $\emptyset$
```

5.2.3 Process: explore

After the agents have colored their neighboring vertices they are ready to explore $S^{**}_{2.5}$ as follows.

[*Informal description*]: Agents move to vertices that need to be guarded or, if they cannot find one, move to vertices that discover more candidates for guarding. For each subtree, agents that do not have children look for neighboring vertices in $\mathrm{D}_{A^C_i}$, prioritizing vertices in G_c. If such a vertex exists, the agent moves to that vertex. If no such vertex exists the agent signals its parent to explore. When an agent moves its parent follows to the vertex that the child was previously occupying.

explore

```
$G = \emptyset$
for $a \in A$ such that $a$.children= $\emptyset$:
    explore_subtree($a$)
for all $a \in A$:
    if $a.child$ moved:
        $a$ moves to last occupied vertex of $a.child$
    $G = G \cup a.v$
    Subfunctions
def explore_subtree($a$):
    if $a$ neighbors $v \in \mathrm{D}_{A^C_i}$ of color $G_c$:
        $v' = v$
    else if $a$ neighbors $v \in \mathrm{D}_{A^C_i}$ of color $1$ or $2$:
        $v' = v$
    else:
        explore_subtree($a.parent$)
        return
    $a$ moves to $v$
    $a$ broadcasts and updates tree
    $a$.explored= $True$
```

Remark 3 (Exploration): As a result of moving to the exploration process, vertices in $S^{*}_{2.5}$ are discovered. These vertices are visible and now belong to $Q(G)$ through completion of the ***2.5D non-redundant peak strategy***. When new vertices are discovered, agents re-evaluate $S^{**}_{2.5}$ and $\mathrm{D}_{A^C_i}$.

5.2.4 Process: merge

After agent exploration, two or more connected components of agents may be able to communicate via neighboring edges. In ***merge***, the exploration sets are updated accordingly and the connected components of agents merge into one larger set.

[*Informal description*]: Let $\{A^C_1, A^C_2, \ldots\}$ denote the set of connected components of agents that are able to merge after exploration, and $A^C_{\#}$ denote the connected component of agents after merging. $\mathrm{K}_{A^C_{\#}}$ contains all vertices in $\mathrm{K}_{A^C_1}, \mathrm{K}_{A^C_2}, \ldots$. $\mathrm{D}_{A^C_{\#}}$ contains all vertices in $\mathrm{D}_{A^C_1}, \mathrm{D}_{A^C_2}, \ldots$ except for any vertex in $\mathrm{K}_{A^C_{\#}}$. Finally, $\mathrm{U}_{A^C_{\#}}$ contains the rest of the vertices, which have not yet been detected by the agents and do not need to be calculated for the algorithm.

merge

```
$\mathrm{K}_{A^C_{\#}} = \emptyset$
$\mathrm{D}_{A^C_{\#}} = \emptyset$
$\{A^C_1, A^C_2, \ldots\}$ is a set of merging connected components
for $A^C_i$ in $\{A^C_1, A^C_2, \ldots\}$
    $\mathrm{K}_{A^C_{\#}} = \mathrm{K}_{A^C_j} \cup \mathrm{K}_{A^C_i}$
for $A^C_i$ in $\{A^C_1, A^C_2, \ldots\}$
    $\mathrm{D}_{A^C_{\#}} = \mathrm{D}_{A^C_j} \cup \mathrm{D}_{A^C_i} \setminus \mathrm{K}_{A^C_j}$
```

5.2.5 Process: redundant_agent_removal

Agents need to place relays in order to maintain connectivity when redundant vertices are found. The process for determining and placing relays is **redundant agent reduction** as described in ***redundant_agent_removal***.

[*Informal description*]: One agent at a time determines if they occupy a redundant vertex by checking if there is a triangle that it uniquely guards (a triangle with vertices not occupied by an agent). If an agent is redundant, it places a relay on its vertex. Relays are considered other agents when it comes to finding roots in ***build_subtrees***Furthermore, if two agents occupy non-redundant vertices that create a doubly redundant edge e_r is detected, one of the two agents colors a vertex $v \in e_r$ to G_c and places a relay on their currently occupied vertex. During the next exploration step, one vertex on e_r will be occupied by either agent. The agent that does not occupy v marks their currently occupied vertex as redundant on the next **redundant agent reduction** step.

redundant_agent_removal

```
for all $a$, asynchronously:
    if if $\Delta^{a.v}_u = 0$
        $a$ places a relay at $a.v$
    if $a$ that uniquely guards a doubly redundant edge
        $a$ places a relay at $a.v$
            $a$ colors a vertex $v$ that is part of the doubly
redundant edge as $G_c$
```

Remark 4 (Tree-based exploration): By construction, the tree-based ordering of agents guarantees $|Q|$ never decreases during exploration because agents are always flowing from the root of the tree to the leafs. The root of the tree by definition contains 2 or more agents. It is necessary that **redundant agent removal** cycles through every agent because the set of redundant agents changes every time a redundant agent is removed.

5.2.6 agent_activation

After **redundant agent removal**, all vertices with color *1* or *2* or vertices with a relay can be considered to be redundant. We use properties of the redundant vertices and their respective redundant edges to add agents to the deployment if necessary.

[*Informal description*]: If a connected component of agents detect a triangle of redundant vertices, 2 agents are added to A^a from A^r.

agent_activation

```
until $|A^a| = \min\left(\frac{3|V_{S^{**}_{2.5}}|-4}{4} + 2|N^r|, \frac{|9V_{S^{**}_{2.5}}|-9}{11}\right)$
    move any $a$ from $A^r$ to $A^a$
```

5.3 2.5D non-redundant peak strategycompletion

In this section we show that our deployment strategy yields a guarding set G^*_{nr} such that $S^{**}_{2.5}$ is fully visible.

Lemma 4 *(Monotonic increase of* $\mathrm{K}_{A^C_i}$ *from* ***explore****): Let* $\mathrm{K}_{A^C_i}$*,* $\mathrm{D}_{A^C_i}$*, and* U *be non-empty for a connected group of agents that contain at least one agent that is a root. Then the cardinality of the set of known vertices* $\mathrm{K}_{A^C_i}$ *monotonically increases as a result of* ***explore****.*

Proof Agents move in ***explore*** when it is their turn based on the hierarchy of the tree that they have formed and there is a suitable neighboring vertex to explore to. As specified in ***explore***, the neighboring vertex always belongs to $\mathrm{D}_{A^C_i}$. This is always possible for some agent because any $v \in \mathrm{D}_{A^C_i}$ is by definition visible by at least one agent. When an agent moves to a vertex $v \in \mathrm{D}_{A^C_i}$, that vertex must become part of the *known* set of vertices $\mathrm{K}_{A^C_i}$, since all triangles touch that vertex will be visible. Because visibility does not change as a result of exploration, $\mathrm{K}_{A^C_i}$ monotonically increases. □

Lemma 5 *(Completion of the* ***2.5D non-redundant peak strategy****): Given* $|A^a| = (3|V_{S^{**}_{2.5}}| - 4)/4$ *and* $|A^r| = (9|V_{S^{**}_{2.5}}| - 9)/11 - |A^a|$ *agents,* ***2.5D non-redundant peak strategy****results in complete exploration of* $S_{2.5}$*.*

Proof Consider a connected group of agents A^C_i and assume $\mathrm{U}_{A^C_i}$ and $\mathrm{K}_{A^C_i}$ are non-empty. Note that there must exist a vertex in $\mathrm{D}_{A^C_i}$ along any path from any two vertices $v_1 \in \mathrm{U}_{A^C_i}$ and $v_2 \in \mathrm{K}_{A^C_i}$. This is because v_1 and v_2 cannot be neighbors of each other, otherwise v_1 would be detected since neighbors of v_2 are visible by definition. Therefore, if $\mathrm{U}_{A^C_i}$ and $\mathrm{K}_{A^C_i}$ are non-empty, $\mathrm{D}_{A^C_i}$ must be non-empty as well. While $\mathrm{K}_{A^C_i}$, $\mathrm{D}_{A^C_i}$, and $\mathrm{U}_{A^C_i}$ are non-empty, Lemma 4 indicates that $|\mathrm{K}_{A^C_i}|$ monotonically increases as long as there are enough agents. Agents in a connected component of agents A^C_i explore until the algorithm is complete or until there are no two agents in A^C_i that occupy the same vertex. In the latter case, the agents remain dormant while waiting for another connected component of agents to make communication during their exploration. We now show that there are enough agents for deployment. As a result of **redundant agent removal**, agents do not occupy redundant vertices and no doubly redundant edges exists. Furthermore, for every redundant triangle that is detected, 2 agents are activated. This is enough because for any 3 redundant vertices that do not form a triangle, there can be at most 2 edges and 3 non-redundant vertices, using $r_{e_r} = 1.5$. If those three redundant vertices do form a triangle, then there are 3 edges and 9/2 non-redundant vertices, using $r_{e_r} = 1.5$. Considering an environment with any number of redundant triangles, the maximum number of non-redundant vertices is given by Theorem 5. Hence, there will always be enough agents in some connected component to explore with

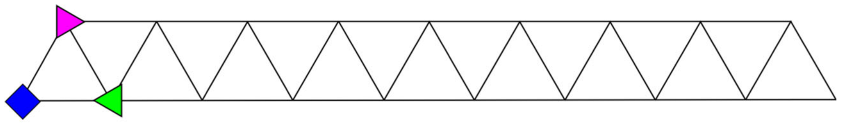
$$|A^a| = \min\left(\frac{3|V_{S^{**}_{2.5}}| - 4}{4} + 2|N^r|, \frac{|9V_{S^{**}_{2.5}}| - 9}{11}\right),$$

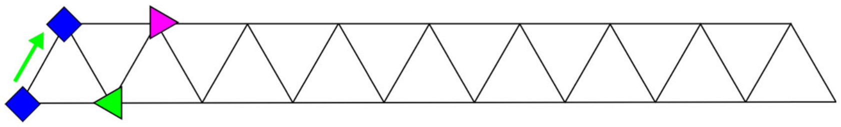
where $\sum_{\forall A^C_i} |A^C_i| = |A^a|$. Note that a vertex cannot belong to two different $\mathrm{K}_{A^C_1}$ and $\mathrm{K}_{A^C_2}$, since otherwise the connected components A^C_1, and A^C_2 would be connected, being part of a single known set. Therefore $\sum_{\forall A^C_i} |\mathrm{K}_{A^C_i}| \leq |V_{S^{**}_{2.5}}|$, where equality only holds when all vertices and faces are visible. Since there is always some connected component with enough agents to explore and given Lemma 4, the collective number of known vertices $\sum_{\forall A^C_i} |\mathrm{K}_{A^C_i}|$ monotonically increases until $\mathrm{U}_{A^C_i}$ and $\mathrm{D}_{A^C_i}$ are empty for all A^C_i or until $\sum_{A^C_i} |\mathrm{K}_{A^C_i}| = |V_{S^{**}_{2.5}}|$. □

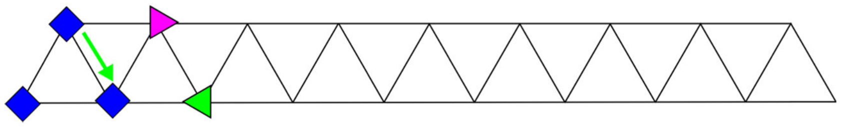
The result of deployment on $S^{**}_{2.5}$ with sufficient number of agents is the occupation of vertices in a guarding set G^*_{nr}. Next, we characterize the number of time steps required by ***2.5D non-redundant peak strategy***for completion.

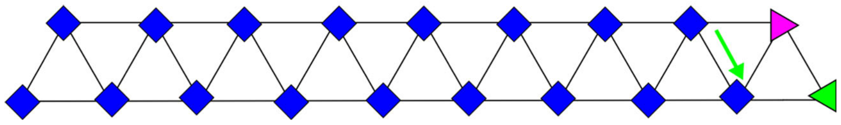
Theorem 7 *(Upper bound on completion time of* ***2.5D non-redundant peak strategy****): The* $S_{2.5}$ *deployment strategy takes at most* $|V_{S^{**}_{2.5}}| - 3$ *time steps to complete.*

Fig. 7 A $S^{**}_{2.5}$ with the potential of taking $|V_{S^{**}_{2.5}}| - 3$ time steps to complete


(a) An initial $S^{**}_{2.5}$ (worst case) after coloring


(b) Second step of $S^{**}_{2.5}$ (worst case) after coloring


(c) Third step $S^{**}_{2.5}$ (worst case) after coloring


(d) Last step of $S^{**}_{2.5}$ (worst case) after coloring

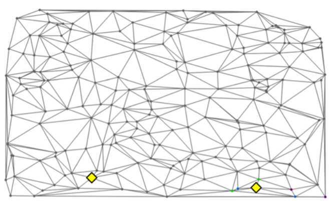
Proof The strategy completes when $Q(G^*_{nr}) = V$, which must be true when $U_{A^C_i} = \emptyset$ and $|K_{A^C_i}| = |V_{S^{**}_{2.5}}|$. From Lemma 4, $|K_{A^C_i}|$ is increased by at least 1 every turn. Therefore, the completion time is dictated by how quickly the agents explore $S^{**}_{2.5}$. Since Q includes v_0, and at least two other vertices (since every vertex has at least two neighbors in the polyhedral terrain), the strategy takes no more than $|V_{S^{**}_{2.5}}| - 3$ time steps to complete. Figure 7 provides an example for this worst-case scenario, showing that the upper bound is attained. □

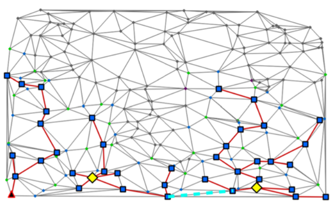
Although it is possible for the algorithm to take up to $|V_{S^{**}_{2.5}}| - 3$ time steps, it is highly unlikely and can be avoided through smarter coloring schemes. In Fig. 7, we use a color scheme that prioritizes labeling by $\{1, 2, G_c\}$. During the exploration phase, there are no unoccupied vertices with color G_c, therefore the agent has to label a vertex with color *1* or *2* to G_c and proceed to move to that vertex.

Next, we characterize a lower bound on completion time of the ***2.5D non-redundant peak strategy***. Let d_o be the max out-degree of vertices in $S^{**}_{2.5}$.

Theorem 8 *(Lower bound on completion time of **2.5D non-redundant peak strategy**): Assume a deployment of **2.5D non-redundant peak strategy**where all agents are initialized on the same vertex. The $S_{2.5}$ deployment strategy takes at least $\frac{|\Delta_{S^{**}_{2.5}}|-2}{d_o-2} - 1$ time steps to complete.*

Proof Upon deployment completion and full 2.5D terrain visibility we know that for each triangular face on the planar graph, an agent occupies a vertex on that triangular face. As agents explore, they create a face-spanning subgraph of


(a) Initialization of the ***2.5D non-redundant peak strategy***


(b) Exploration and coloring after 14 time steps step

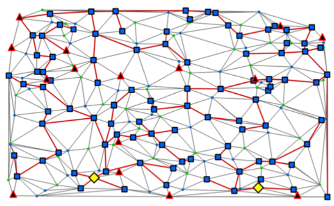

(c) Fully guarded environment

Fig. 8 Execution of ***2.5D non-redundant peak strategy***on $S^{**}_{2.5}$ with 200 vertices

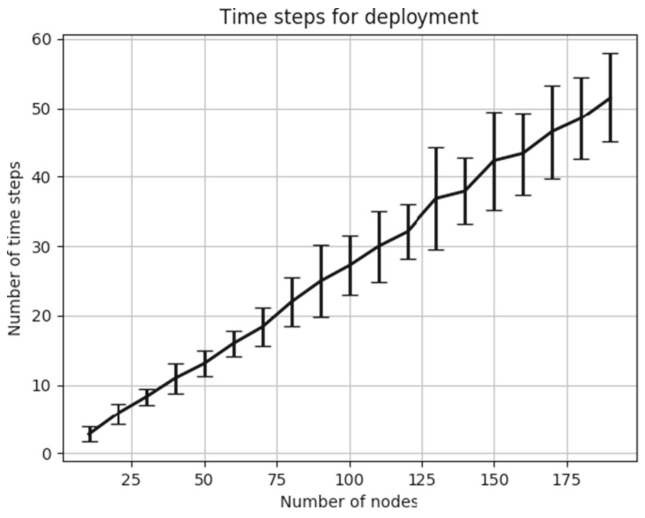


Fig. 9 Number of time steps required to complete deployment using the ***2.5D non-redundant peak strategy*** versus the number of nodes in the environment. Each data point is averaged over 100 trials. The error bars show the corresponding standard deviation

the complete environment that consists of vertices occupied by relays and agents. A face-spanning subgraph can be created with no less than $\frac{\Delta_{S_{2.5}^{**}}-2}{d_o-2} - 1$ vertices [20]. Lastly, from Lemma 4 we have that ***2.5D non-redundant peak strategy*** expands the face-spanning subgraph by 1 vertex per step. □

6 Implementation

To illustrate the practicality of our algorithm, we provide here both simulation and experimental results on group of robots.

Figure 8 shows several steps of the simulation of ***2.5D non-redundant peak strategy***on a 2.5D terrain. The terrain has 200 nodes. In (a), 149 and 14 agents are initialized in A^a and A^r, respectively, at either one of two initial nodes, which become roots represented by . Roots occur when 2 or more agents occupy the same vertex. The vertices that are visible to the agents are denoted with either blue or green dots. The agents label the newly detected vertices with colors priority: *1* : green dot, *2* : red dot, and G_c : . Agents from each one of the starting nodes will explore unaware of the other group of agents and their node explorations until they make communication visually by occupying adjacent nodes. Once the agents are able to communicate with the other group, cohesion is never lost as agents keep exploring. In (b), agents have explored and 14 time steps have passed. At this point, two connected components of agents are able to communicate via the dashed edge shown on the bottom of the image. Hence, ***merge***combines the exploration sets of each of the connected components so knowledge of the environment is shared. There is also a vertex shown as shown in the bottom left. This vertex was deemed redundant by agents so a relay is placed here. In (c) the environment is completely visible and the deployment is complete after 55 time steps.

In Fig. 9, we display the number of time steps required for the agents to complete deployment on the 2.5D terrain as a function of the number of nodes in the environment. As one can see from the plot, the required number of time steps increases roughly linearly with the number of nodes. Noting this fact, the computational complexity of the average case looks linear, since the amount of time it takes an agent to compute its step is independent of the total number of nodes in the environment (and only depends on its neighboring nodes).

We have also implemented ***2.5D non-redundant peak strategy***on a group of robots in the Robotarium [21]. The Robotarium is a multi-agent platform for remotely testing swarm robot applications. In this experiment, 12 agents are deployed on $S_{2.5}^{**}$ with 24 vertices. In order to deploy, the agents use controllers based on their unicycle dynamics to reach vertices that they explore, which they treat as waypoints. Agents initially begin at the vertex shown in Fig. 11a and find vertices to guard shown in Fig. 11b. At every time step, agents are given a set amount of time to reach their waypoints before computing other steps of the algorithm, such as coloring and vertex redundancy. Because the platform utilizes robots for 2D environments, we projected the 2.5D

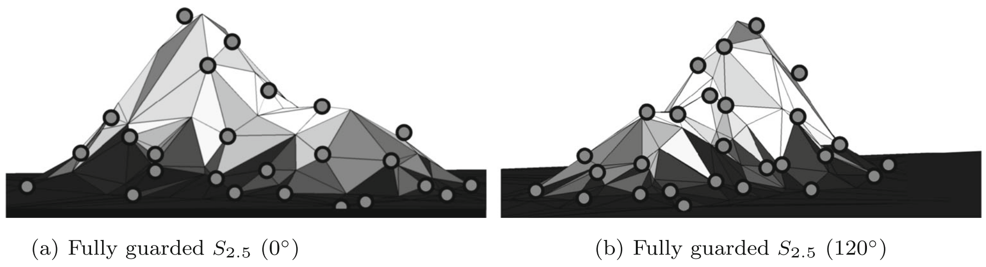

(a) Fully guarded $S_{2.5}$ (0°)
(b) Fully guarded $S_{2.5}$ (120°)

Fig. 10 For agent deployment, $S_{2.5}$ is transformed into a planar graph $S_{2.5}^{**}$. The guarding set G_{nr}^* as a result of ***2.5D non-redundant peak strategy***on $S_{2.5}^{**}$ from Fig. 8 shown as large dots on the environment. The respective vertices in the 2.5D environment $S_{2.5}$ are shown at different angles

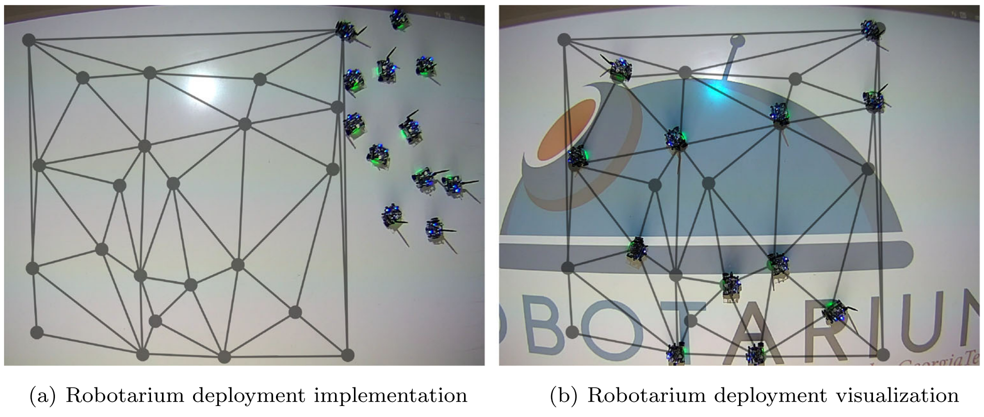

(a) Robotarium deployment implementation
(b) Robotarium deployment visualization

Fig. 11 Deployment of ***2.5D non-redundant peak strategy*** using Robotarium [21]. A projection of the $S_{2.5}$ to a planar graph $S_{2.5}^{**}$ overlays the images. In (a) agents are shown at their initial location at the vertex on the top right. To avoid collisions, agents start too the side of the vertex. The final configuration of the agents are shown in (b) where the environment is fully visible

environment to a planar graph which is shown as an overlay in Fig. 11b.

7 Conclusions

We have designed coordination algorithms for distributed exploration and guarding of agents in 1.5D and 2.5D environments. For 1.5D settings, we have determined a number of active agents that, on average, is sufficient to achieve full visibility and provided enough reserve agents when the ratio of peaks to valleys is high. We have developed the ***1.5D alternate peak strategy***, established that it leads to agent deployment with full visibility of the environment, and characterized its time of completion. For 2.5D settings, we have characterized the necessary number of agents required to achieve visibility for environments with and without embedded triangles. We have introduced the ***2.5D non-redundant peak strategy***, which allows for groups of agents to begin on different nodes and operate independently from each other. These groups dynamically create subtrees and choose to explore nodes that maintain connectivity. Separate groups eventually meet, merge their learned information about the environment and deployment in a consistent way, and continue exploring as a new single group. We show that this process may be repeated until the overall group of agents achieves full visibility, and provide both upper and lower bounds on the time of completion. Future work will explore scenarios for agents with limited visibility and anisotropic visibility regions. A possible line of attack to accommodate range-limited agent visibility would have the algorithm add vertices to the 1.5D and 2.5D terrains to limit inter-node distances. Furthermore, future work will extend the analysis to asynchronous algorithm executions, and incorporate increasingly realistic considerations in modeling the interactions of the agents with the physical world, such as agents traversing across the planes of the 2.5D terrain instead of just the edges and accounting for agent dynamics in the algorithm execution. Finally, the proposed algorithm is conservative in that it has agents reasoning over a convex subset of their full visibility region. We are also interested in improving the algorithm efficiency by employing the full agents' visibility regions.

Acknowledgments This work was supported by ONR Award N00014-16-1-2836.

References

1. Almadhoun, R., Taha, T., Seneviratne, L., Zweiri, Y.: A survey on multi-robot coverage path planning for model reconstruction and mapping. SN Applied Sciences **1**(8), 847 (2019)
2. Appel, K., Haken, W.: Every planar map is four colorable. Part i: Discharging. Illinois J. Math **21**, 429–490 (1977)
3. Ashok, P., Fomin, F.V., Kolay, S., Saurabh, S., Zehavi, M.: Exact algorithms for terrain guarding. ACM Transactions on Algorithms (T,ALG) **14**(2), 1–20 (2018)
4. Barenboim, L., Elkin, M.: Distributed graph coloring: Fundamentals and recent developments. Synthesis Lectures on Distributed Computing Theory **4**(1), 1–171 (2013)
5. de Berg, M., van Kreveld, M., Overmars, M., Schwarzkopf, O.: Computational geometry: Algorithms and Applications, 2 edn, Springer (2000)
6. Bose, P., Shermer, T., Toussaint, G., Zhu, B.: Guarding polyhedral terrains. Comput. Geom. **7**, 173–185 (1997)
7. Bullo, F., Cortés, J., Martinez, S.: Distributed control of robotic networks. Applied mathematics series. Princeton University Press (2009)
8. Chiba, N., Nishizeki, T., Saito, N.: A linear 5-coloring algorithm of planar graphs. Journal of Algorithms **2**, 317–327 (1981)
9. Chvátal, V.: A combinatorial theorem in plane geometry. Journal of Combinatorial Theory. Series B **18**, 39–41 (1975)
10. Friedrichs, S., Hemmer, M., Schmidt, C.: A PTAS for the Continuous 1.5D Terrain Guarding Problem. In: Canadian Conference on Computational Geometry. Halifax, Nova Scotia, Canada, Electronic Proceedings (2014)
11. Ganguli, A., Cortés, J., Bullo, F.: Distributed deployment of asynchronous guards in art galleries. In: American Control Conference, pp. 1416–1421, Minneapolis (2006)
12. Ghaffari, M., Lymouri, C.: Simple and near-optimal distributed coloring for sparse graphs. arXiv:1708.06275 (2017)
13. Gibson, M., Kanade, G., Krohn, E., Varadarajan, K.: Guarding terrains via local search. Journal of Computational Geometry **5**(1), 168–178 (2014)

14. Hurtado, F., Loffler, M., Matos, I., Sacristan, V., Saumell, M., Silveria, R.I., Staals, F.: Terrain visibility with multiple viewpoints. In: International Symposium on Algorithms and Computation, pp. 317–327, Jeonju, Korea (2014)
15. Khodakarami, F., Didehvar, F., Mohades, A.: 1.5 d terrain guarding problem parameterized by guard range. Theor. Comput. Sci. **661**, 65–69 (2017)
16. Ma, A., Cortés, J.: Visibility-based distributed deployment of robotic teams in polyhedral terrains. In: ASME Dynamic Systems and Control Conference. Minneapolis, MN, DSCC2016-9820 (2016)
17. Maini, P., Gupta, G., Tokekar, P., Sujit, P.: Visibility-based monitoring of a path using a heterogeneous robot team. In: IEEE/RSJ Int. Conf. On Intelligent Robots & Systems, pp. 3765–3770 (2018)
18. Mesbahi, M., Egerstedt, M.: Graph theoretic methods in multiagent networks. Applied mathematics series. Princeton University Press (2010)
19. O'Rourke, J.: Art gallery theorems and algorithms. Oxford University Press (1987)
20. Patwary, M., Rahman, S.: Minimum face-spanning subgraphs of plane graphs. AKCE International Journal of Graphs and Combinatorics **7**(2), 133–150 (2010)
21. Pickem, D., Glotfelter, P., Wang, L., Mote, M., Ames, A., Feron, E., Egerstedt, M.: The Robotarium: A remotely accessible swarm robotics research testbed. In: IEEE Int. Conf. On Robotics and Automation, pp. 1699–1706. Singapore (2017)
22. Robertson, N., Sanders, D.P., Seymour, P., Thomas, R.: Efficiently four-coloring planar graphs. In: ACM Symposium on Theory of Computing, pp. 571–575, Philadelphia (1996)
23. Savkin, A.V., Huang, H.: Proactive deployment of aerial drones for coverage over very uneven terrains: a version of the 3d art gallery problem. Sensors **19**(6), 1438 (2019)
24. Shermer, T.C.: Recent results in art galleries. Proc. IEEE **80**(9), 1384–1399 (1992)
25. Veenstra, K., Obraczka, K.: Guiding Sensor-Node Deployment over 2.5D Terrain, In: 2015 IEEE International Conference on Communications (ICC), pp. 6719–6725 (2015)
26. Williams, M.H.: A linear algorithm for colouring planar graphs with five colours. Comput. J. **28**, 78–81 (1985)

Aaron Ma is an AI engineer at Shield AI who is interested in the application of reinforcement learning in multi-agent systems. He received his Masters and Ph.D. in Mechanical and Aerospace Engineering from University of California, San Diego from 2014-2020 under the direction of Prof. Jorge Cortes. His research interests include multi-agent robotics, machine learning, and control systems.

Jorge Cortés received the Licenciatura degree in mathematics from Universidad de Zaragoza, Zaragoza, Spain, in 1997, and the Ph.D. degree in engineering mathematics from Universidad Carlos III de Madrid, Madrid, Spain, in 2001. He held postdoctoral positions with the University of Twente, Twente, The Netherlands, and the University of Illinois at Urbana-Champaign, Urbana, IL, USA. He was an Assistant Professor with the Department of Applied Mathematics and Statistics, University of California, Santa Cruz, CA, USA, from 2004 to 2007. He is currently a Professor in the Department of Mechanical and Aerospace Engineering, University of California, San Diego, CA, USA. He is the author of Geometric, Control and Numerical Aspects of Nonholonomic Systems (Springer-Verlag, 2002) and co-author (together with F. Bullo and S. Martínez) of Distributed Control of Robotic Networks (Princeton University Press, 2009). He is a Fellow of IEEE and SIAM. His current research interests include distributed control and optimization, network science, opportunistic state-triggered control, reasoning and decision making under uncertainty, and distributed coordination in power networks, robotics, and transportation.