Exploiting Robotic Swarm Characteristics for Adversarial Subversion in Coverage Tasks

Navyata Sanghvi

CMU-RI-TR-17-60

August 2017

Submitted in partial fulfillment of the requirements for the degree of Master of Science in Robotics

Robotics Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

Thesis Committee:
Katia Sycara, Chair
Maxim Likhachev
Sasanka Nagavalli

For my parents

Abstract

Multi-robot systems, such as swarms, with large number of members that are homogeneous and anonymous are robust to deletion and addition of members. However, these same properties that make the system robust, create vulnerabilities under certain circumstances. In this work, we study such a case, namely the insertion by adversarial agents, called moles, that subvert the performance of the system. The adversary monitors the swarm's movements during surveillance operations for the presence of holes, i.e. areas that were left uncovered by the swarm. The adversary then adds moles that get positioned in the swarm, in such a way as to deceive the swarms regarding the existence of holes and thus preventing the swarm from discovering and repairing the holes. This problem has significant military applications. Our contributions are as follows: First, to the best of our knowledge, this is the first paper that studies this problem. Second, we provide a formalization of the problem. Third, we provide several algorithms, and characterize them formally and also experimentally. Finally, based on developed theory and algorithms, we present a dynamic scenario and describe adversary control laws to leverage the identified swarm vulnerability.

Acknowledgements

I would like to begin by thanking my advisor, Professor Katia Sycara, for her unerring guidance and enthusiasm. Her passion for the field of swarm robotics, as well as her constant advice and insights have created for me a novel, exciting and highly interesting journey in robotics research. Through all the triumphs and failures accompanying this foray, I am honored to have had her vast knowledge and invariable support by my side.

I would also like to thank Sasanka Nagavalli, whose experience and keen intellect have both been an inspiration as well as a steadying hand. He has spent countless hours with me on insightful discussions which have helped me overcome many a stressful time and have always steered my research in new, fulfilling directions.

I would like to thank my committee member Professor Maxim Likhachev for interesting questions and comments, and the opportunity to have his experienced judgment to critique and better my work. I would like to thank Professor Nilanjan Chakraborty, Professor Michael Lewis and Changjoo Nam for their expert advice during many lab meetings. I would also like to thank other members of our lab, Wenhao Luo, Shehzaman Khatib, Anqi Li, Jimit Gandhi, and Ramitha Sundar for many conversations, lots of help, laughter and sharing-in-strife during both study and work. It has been an invaluable experience to work, learn and grow with this bunch.

Finally, I would like to express my endless thanks to my family and parents for their many sacrifices, long-distance support, love, cheers, and boundless faith.

Contents

1 Introduction

In recent years, there has been significant interest in distributed multi-robot systems whose members act based on information acquired through local sensing and/or communication with other robots in their spatial neighborhood. When these local interactions result in global collective behaviors (e.g. rendezvous, flocking, dispersion), the system is known as a robotic swarm [3, 11]. Robotic swarms are composed of a large number of agents that are homogeneous. Additionally, swarms are robust to addition or subtraction of agents, which gives them the beneficial properties of scalability and robustness to individual robot failure. However, these same characteristics also make the swarm vulnerable to manipulation by agents that could be inserted into the swarm by an adversary for the purpose of subverting the swarm's performance.

Swarms have great potential for many applications including search and rescue, environmental monitoring, exploration, reconnaissance and surveillance. Particularly for military applications, they have the potential to be an excellent asset when employed by allied forces or a dangerous threat when used by enemies. Robotic swarms are envisioned to be composed of relatively inexpensive (even disposable) robots such as commercially available quadrotors that only cost tens or hundreds of dollars, which makes them easily accessible to many parties — even those with limited monetary resources. Based on current trends [16], [14], it is reasonable to expect that robotic swarms will continue to decrease in cost. Thus, it has been argued [17] in the military literature that when defending against a hostile swarm, it is not cost-effective to use traditional means to disrupt or destroy the swarm (e.g. ammunition expended to destroy swarm may be more expensive than swarm itself). In these situations, a better strategy would be to exploit swarm vulnerabilities to insert adversarial agents for the purpose of disrupting swarm performance, with the added advantage that the enemy still thinks its swarm works correctly. Studying such deception strategies is also necessary to guide development of counter-measures against disruption of swarm behavior by adversarial agents in friendly swarms.

In this work, we study a scenario in which a hostile swarm is performing surveillance operations. Each robot in the swarm is assumed to have range-limited communication and sensing. A point in the environment is considered "covered" for surveillance if it is within the sensing disk of any swarm robot. As the swarm robots slowly move ensuring that they maintain communication connectivity, holes in coverage dynamically appear. The detection of "coverage" holes in sensor networks using both topological approaches [9] and metric approaches [20, 12] has been studied in the literature [1]. One approach based on localized Voronoi diagrams [22] requires only local information and one-hop communication between members in the sensor network, so it is particularly applicable to swarms and we assume the swarm under study uses this approach. By monitoring the swarm's movements, our goal is to periodically identify the *number* and *location* where adversarial "mole" agents must be inserted into the hostile swarm to prevent its original "citizen" agents from detecting any holes in coverage.

This research makes the following contributions. First, to the best of our knowledge, this is the first work that studies this problem. Second, we provide a formalization of the problem. Third, we provide several algorithms, and characterize them formally and also experimentally. In Section 3, we formalize this problem. In Section 4, we present and characterize several algorithms to find the required number of moles and their insertion locations. In Section 5, we present simulation results and discuss the effectiveness of each algorithm. Finally, in Section 6 we describe a dynamic scenario and adversary control laws based on the developed theory and algorithms.

2 Related Work

The problem of coverage hole discovery in sensor networks has been widely studied in the literature [1, 21]. A variety of approaches to coverage hole discovery have been considered for both static and mobile networks including those based on computational geometry [22, 12] and topology [9, 8]. Computational geometry approaches typically involve computing the Voronoi diagram [2] and then (for mobile sensor networks) following a simple motion rule to heuristically minimize coverage holes [20] (e.g. moving towards the furthest Voronoi vertex, minimizing the maximum distance of any point to the nearest Voronoi vertex). Topological approaches make minimal metric assumptions (e.g. the ability to distinguish 'near' and 'far' neighbors) and then use only information based on the connectivity of the sensing graphs to find and repair holes [7]. While the previous literature has studied hole discovery, we study the novel problem of how to place agents to *prevent* hole discovery.

While there is significant literature on security in sensor networks [4, 18], that work has focused on security in communications based on (a) attacks on secrecy and authentication (e.g. unauthorized snooping on private communication channels), (b) attacks on network availability (e.g. overloading the network to cause distributed denial of service) or (c) attacks on service integrity (e.g. compromising a sensor in the network and injecting false data). That work is not relevant to our work since we do not try to compromise network security, availability or integrity through communications. Instead, we exploit physical vulnerabilities intrinsic to robotic swarms in order to insert the minimum number of mole agents to prevent the hostile swarm from successfully detecting coverage holes.

Previous work on Particle Swarm Optimization (PSO) [15] defines 'deception' as the average proportion of optimization iterations in which the selected and true neighborhood bests are different due to noise in particles' personal best objective values, leading to sub-optimal particle propagation. The actions, beliefs and deception strategies of a deceiver robot against its mark have also been studied using game theory [19], but in the context of two individuals' interaction. In contrast to such prior work on robotic deception, we are the first to form a deliberate, structured attack on a citizen swarm through adversarial agent insertion to subvert its performance. We demonstrate this attack in the context of a swarm performing surveillance operations.

3 Problem Formulation

3.1 Preliminaries

We present in this sub-section preliminary concepts, definitions and assumptions used to define our problem.

Citizen and Mole Agents: Assume the hostile swarm is composed of n *citizen agents* $\mathcal{P} = \{\mathbf{p}_1, \mathbf{p}_2, ..., \mathbf{p}_n\}$, $\mathbf{p}_i \in \mathbb{R}^2$. We wish to insert a set of adversarial *mole agents* $\mathcal{Q} = \{\mathbf{q}_1, \mathbf{q}_2, ...\}$, $\mathbf{q}_i \in \mathbb{R}^2$ to disrupt the performance of the hostile swarm. Our goal is to identify the quantity and locations of moles necessary to disrupt swarm performance. We assume moles communicate identically to citizens. That is, messages are exchanged between moles in the same manner and format as between citizens.

Sensing, Communication and Agent Radius: We assume the sensing range r_s and communication range r_c are identical for all agents $(\mathcal{P} \cup \mathcal{Q})$ (i.e. $r_s = r_c = r$). In addition, we assume each agent occupies a disk of radius r_{min}. That is, $\forall \mathbf{u}_i, \mathbf{u}_j \in (\mathcal{P} \cup \mathcal{Q}) : \|\mathbf{u}_i - \mathbf{u}_j\|_2 \geq 2r_{min}$. To make connectivity possible, $r_{min} \in \left(0, \frac{r}{2}\right]$. We assume the citizen swarm is connected. When adding moles, we must ensure that the resulting network of agents $(\mathcal{P} \cup \mathcal{Q})$ is connected. If mole agents $\mathbf{q}_i$ are added incrementally one-by-one, the following condition is both necessary and sufficient to ensure global connectivity: $\exists \mathbf{v} \in (\mathcal{P} \cup \mathcal{Q}) : \|\mathbf{q}_i - \mathbf{v}\|_2 \leq r$.

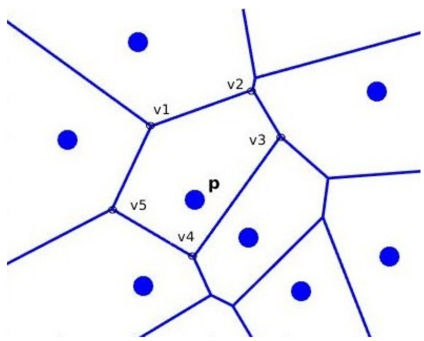


Figure 1: Voronoi cells of 8 agents. Agent $\mathbf{p}$'s cell is $\square v_1 v_2 v_3 v_4 v_5$.

Voronoi Partition [2]: Consider the $\mathbb{R}^2$ plane. Given the set of citizen agent locations $\mathcal{P} \subset \mathbb{R}^2$, a Voronoi partition of the plane divides it into convex polygons known as Voronoi cells (one per agent). A Voronoi cell corresponding to agent $\mathbf{p} \in \mathcal{P}$ is the set of all points closer to $\mathbf{p}$ than to any other agent. Formally, if $H(\mathbf{p}_i, \mathbf{p}_j)$ represents the half-plane defined by the perpendicular bisector of the line segment joining agents $\mathbf{p}_i, \mathbf{p}_j \in \mathcal{P}$ and containing $\mathbf{p}_i$, the Voronoi cell of $\mathbf{p}_i$ is given by the following.

$$cell_{vd}(\mathbf{p}_i) = \bigcap_{\mathbf{p}_j \in \mathcal{P} \setminus \{\mathbf{p}_i\}} H(\mathbf{p}_i, \mathbf{p}_j) \tag{1}$$

Voronoi Vertices: We represent the Voronoi cell corresponding to agent $\mathbf{p} \in \mathcal{P}$ by its set of vertices $\mathcal{V}_{\mathbf{p}}(\mathcal{P})$ generated by the Voronoi partition of $\mathcal{P}$. Figure 1 shows an example of a Voronoi partition where, for agent $\mathbf{p}$, $\mathcal{V}_{\mathbf{p}}(\mathcal{P}) = \{v_1, v_2, v_3, v_4, v_5\}$.

Boundary Agents: When the swarm includes both citizens $\mathcal{P}$ and moles $\mathcal{Q}$, *boundary agents* $\mathcal{B}_{\mathcal{Q}}(\mathcal{P}) \subseteq \mathcal{P}$ are the subset of citizens at the edge of a coverage hole. Since Voronoi cells are defined as the set of points closest to the corresponding agent, if any point is within an agent's Voronoi cell but outside its sensing range, that point is not within *any* agent's sensing range (i.e. there is a coverage hole and this agent is at the boundary of that coverage hole). Since Voronoi cells are convex, a boundary agent is any citizen with one or more Voronoi vertices outside of its sensing range r.

$$\mathcal{B}_{\mathcal{Q}}(\mathcal{P}) = \{\mathbf{p} \in \mathcal{P} | \exists \mathbf{v} \in \mathcal{V}_{\mathbf{p}}(\mathcal{P} \cup \mathcal{Q}) : \|\mathbf{v} - \mathbf{p}\|_2 > r\} \tag{2}$$

Internal Agents: These are the subset of citizen agents $\mathcal{I}_{\mathcal{Q}}(\mathcal{P}) \subseteq \mathcal{P}$ which are not boundary agents (i.e. $\mathcal{I}_{\mathcal{Q}}(\mathcal{P}) = \mathcal{P} \setminus \mathcal{B}_{\mathcal{Q}}(\mathcal{P})$).

3.2 Problem Statement

Given a multi-agent configuration of citizen agents $\mathcal{P}$ performing an exploration and surveillance mission, our problem is identifying near-optimally the number and insertion locations of mole agents $\mathcal{Q}$ to prevent the discovery of coverage holes in a given state of the citizen system. These insertion points are identified such that each citizen agent (after the insertion) believes that it is *not* on the edge of a hole (i.e. it is *not* a boundary agent) and thus, that the system as a whole has achieved full coverage.

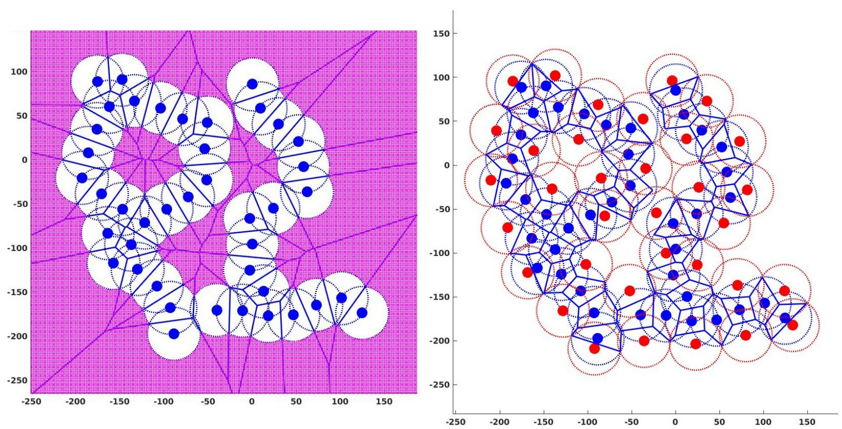


Figure 2: In this figure, circles represent the sensing disk of each agent with radii equal to sensing ranges r, and solid disks (blue or red) represent the physical area occupied by each agent with radii = r_{min}. The left panel shows the citizen agent formation in $\mathbb{R}^2$ and their Voronoi cells. The area in pink represents holes in sensing. The right hand panel displays an example solution including citizens (in blue), moles (in red), and mole agent placements and sensing disks (also in $\mathbb{R}^2$). The placement of the moles ensures that no citizen is a boundary agent. This can be seen from the resulting Voronoi cells of citizens, shown via blue lines.

Consider unweighted undirected graph $G = (V, E)$ with vertex set $V = \mathcal{P} \cup \mathcal{Q}$ (i.e. citizens and moles are vertices) and edges $E = \{(i, j) \mid \mathbf{v}_i, \mathbf{v}_j \in V : \|\mathbf{v}_i - \mathbf{v}_j\|_2 \leq r\}$ (i.e. an edge connects two agents if they are within sensing range of each other). Let the Laplacian matrix for this graph be given by $L(G)$. It is known that the eigenvalues of this matrix are non-negative (i.e. $\forall k : 0 \leq \lambda_k \leq \lambda_{k+1}$). In addition, the second smallest eigenvalue is non-zero (i.e. $\lambda_2 > 0$) if and only if the graph is connected [13].

We must ensure this condition is true to ensure global connectivity of our network of agents. If mole agents are incrementally inserted, incorporating sensing and agent radius constraints from Section 3.1 gives us our objective:

$$
\begin{array}{ll}
\underset{\mathcal{Q}}{\arg\min} & |\mathcal{Q}| \\
\text{subject to} & \mathcal{B}_{\mathcal{Q}}(\mathcal{P}) = \varnothing \\
\forall \mathbf{u}_i, \mathbf{u}_j \in (\mathcal{P} \cup \mathcal{Q}) & : \|\mathbf{u}_i - \mathbf{u}_j\|_2 \geq 2r_{min} \\
\lambda\left(L(G)\right) = \left\{\lambda_1, \lambda_2, \ldots, \lambda_{|\mathcal{P}\cup\mathcal{Q}|}\right\} & : \lambda_2 > 0
\end{array}
$$

In Figure 2, an example of a formation of citizen agents is shown in the left panel, along with an example solution (i.e. the near-optimal insertion locations of mole agents) in red in the right panel. Also shown are the *resulting* Voronoi cells of the citizen agents after mole insertion, which are all within the respective agents' sensing ranges, thus making the system of citizen agents conclude that there are no holes in sensing.

4 Algorithms

4.1 Preliminaries

We introduce here some preliminary terminology, definitions and theorems required to understand our algorithms.

4.1.1 Terminology

To illustrate the following concepts, consider the example boundary citizen agent in Figure 3. We assume that the plane contains citizens $\mathcal{P}$ and moles $\mathcal{Q}$.

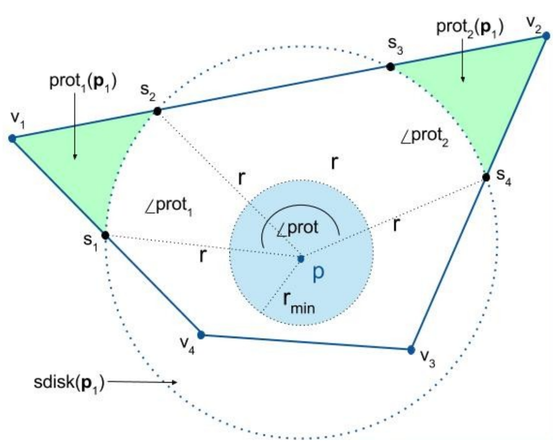


Figure 3: Example of a citizen agent $\mathbf{p}_1$, its Voronoi cell $\square v_1 v_2 v_3 v_4$, sensing disk $sdisk(\mathbf{p}_1)$, protrusions $prot_k(\mathbf{p}_1)$, angle of protrusion $\angle prot$ and pieced angles of protrusion $\angle prot_k$ for $k \in \{1, 2\}$.

Definition 1. *Protrusions* $prot_k(\mathbf{p}),\ \ k \in \{1, 2, .., K\}$ from a boundary agent $\mathbf{p}$'s sensing disk $sdisk(\mathbf{p})$ are disjoint regions in its Voronoi cell $cell_{vd}(\mathbf{p})$ that are outside of its sensing range r.

Definition 2. A boundary agent $\mathbf{p}$'s *protrusion angles* $\angle prot_k$, $k \in \{1, 2, ..., K\}$ are angles subtended by each protrusion $prot_k(\mathbf{p})$ at its center.

Definition 3. A boundary agent $\mathbf{p}$'s *total protrusion angle* $\angle prot_{tot}$ is the minimum total angle subtended by all protrusions $prot_k(\mathbf{p})$ at its center. Note that $\angle prot_{tot} \geq \sum\limits_k \angle prot_k$.

Definition 4. A boundary agent $\mathbf{p}$'s *protrusion angle* $\angle prot_k$ is said to be *flipped* by the insertion of mole agents $\mathcal{Q}_{ins}$ iff all points in $prot_k(\mathbf{p})$ are in the Voronoi cell of one $\mathbf{q} \in \mathcal{Q}_{ins}$ after insertion.

Definition 5. A boundary agent $\mathbf{p}$'s *total protrusion angle* $\angle prot_{tot}$ is said to be *flipped* by the insertion of mole agents $\mathcal{Q}_{ins}$ iff all its protrusion angles are flipped by the insertion. Using Definition 1, this is equivalent to saying that $\mathbf{p}$ is no longer a boundary agent.

Definition 6. *A boundary agent* $\mathbf{p}$ is said to be *flipped* by the incremental insertion of mole agents $\mathcal{Q}_{ins}$ iff its protrusion angle $\angle prot_{tot}$ has been flipped by the insertion.

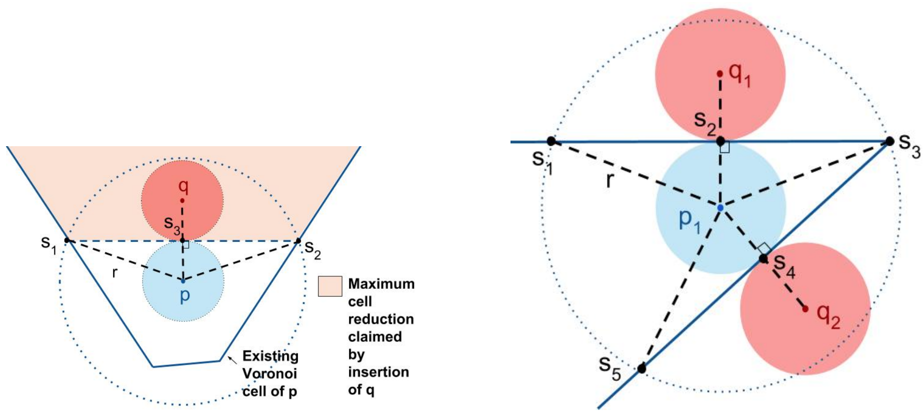
4.1.2 Mole Agents per Citizen Agent

Here we prove that the number of moles required to flip any citizen boundary agent is bounded.

Lemma 1. *A protrusion angle $\angle prot_k$ can be flipped by the insertion of a single mole iff:*

$$\angle prot_k \leq 2\cos^{-1}\left(\tfrac{r_{min}}{r}\right) \tag{3}$$

Proof. Given any boundary citizen $\mathbf{p} \in \mathcal{B}_{\mathcal{Q}}(\mathcal{P})$ and its Voronoi cell, a new mole $\mathbf{q}$ may be placed anywhere within $\mathbf{p}$'s sensing range r such that $\forall \mathbf{u} \in (\mathcal{P} \cup \mathcal{Q})$: $\|\mathbf{q} - \mathbf{u}\|_2 \geq 2r_{min}$. The maximum reduction in size of Voronoi cell of $\mathbf{p}$ is achieved when $\mathbf{q}$ is placed as close as possible to $\mathbf{p}$ (i.e. $\|\mathbf{q} - \mathbf{p}\|_2 = 2r_{min}$). Consider Figure 4a. From congruent triangles $\triangle s_1\mathbf{p}s_3$ and $\triangle s_3\mathbf{p}s_2$, this theorem must be true for $\angle prot_k = \angle s_1\mathbf{p}s_2 = 2\cos^{-1}\left(\frac{r_{min}}{r}\right)$. The insertion would result in new Voronoi edge $\overline{s_1 s_2}$. A smaller protrusion angle, i.e., $\angle prot_k \leq 2\cos^{-1}\left(\frac{r_{min}}{r}\right)$ would also require inserting only one mole to flip the citizen, but it would not require $\|\mathbf{q} - \mathbf{p}\|_2 = 2r_{min}$. A larger protrusion angle, i.e., $\angle prot_k > 2\cos^{-1}\left(\frac{r_{min}}{r}\right)$ would require more than one mole to flip. Hence, proved that a single mole agent insertion can flip $\angle prot_k$ iff $\angle prot_k \leq 2\cos^{-1}\left(\frac{r_{min}}{r}\right)$. □



(a) A citizen boundary agent $\mathbf{p}$, its Voronoi cell before mole insertion, the closest mole insertion location $\mathbf{q}$ ($2r_{min}$ from $\mathbf{p}$), and the maximum half-plane claimed by the insertion, flipping $\mathbf{p}$.

(b) Illustration that when there is only one citizen agent $\mathbf{p}_1$, three moles are required to flip it.

Figure 4: Mole agent insertion.

Theorem 1. *When there is only one citizen, at most three moles are required to flip it.*

Proof. First, we show that no configuration of one or two moles can flip the citizen. When we have only one citizen $\mathbf{p}_1$, its Voronoi cell is the entire plane and it has one protrusion angle $\angle prot_1 = \angle prot_{tot} = 2\pi$. From Lemma 1, adding one mole $\mathbf{q}_1$ cannot decrease the Voronoi cell of $\mathbf{p}_1$ such that it becomes an internal agent, since $r_{min} \in (0, \frac{r}{2}] \Rightarrow \angle prot_1 > 2\cos^{-1}\left(\frac{r_{min}}{r}\right)$. After insertion of $\mathbf{q}_1$, the resulting $\angle prot_1$ is minimized when $\mathbf{q}_1$ is placed $2r_{min}$ away from $\mathbf{p}_1$. That is, $\angle prot_1 \geq (2\pi - 2\cos^{-1}\left(\frac{r_{min}}{r}\right)) \in (\pi, \frac{4\pi}{3}] \Rightarrow \angle prot_1 > \pi$. This cannot be flipped by one more mole $\mathbf{q}_2$, since, from Lemma 1, $2\cos^{-1}\left(\frac{r_{min}}{r}\right) \in [\frac{2\pi}{3}, \pi) \Rightarrow \angle prot_1 > 2\cos^{-1}\left(\frac{r_{min}}{r}\right)$. Therefore, two moles cannot flip a citizen in this scenario.

We now prove that three moles are sufficient to flip this citizen. Clearly, it is sufficient to show one configuration of the three which makes flipping possible. Consider Figure 4b, one arrangement of $\mathbf{p}_1$, $\mathbf{q}_1$ and $\mathbf{q}_2$. In this case, both $\mathbf{q}_1$ and $\mathbf{q}_2$ are placed at $2r_{min}$ from $\mathbf{p}_1$. They are arranged such that $\angle s_1\mathbf{p}_1 s_2 = \angle s_2\mathbf{p}_1 s_3 = \angle s_3\mathbf{p}_1 s_4 = \angle s_4\mathbf{p}_1 s_5 = \cos^{-1}\left(\frac{r_{min}}{r}\right)$. The resulting angle of protrusion is $\angle s_1\mathbf{p}_1 s_5 = 2\pi$ - $4\cos^{-1}\left(\frac{r_{min}}{r}\right)$. Applying Lemma 1, for another mole $\mathbf{q}_3$ to flip this citizen, we need 2π - $4\cos^{-1}\left(\frac{r_{min}}{r}\right) \leq 2\cos^{-1}\left(\frac{r_{min}}{r}\right)$. This is true whenever $r_{min} \leq \frac{r}{2}$, which is always true since $r_{min} \in \left(0, \frac{r}{2}\right]$. Therefore, three moles are required to flip a single citizen agent. $\square$

Applying arguments from Theorem 1, it is clear that a *maximum* of three moles are required to flip any boundary citizen.

4.1.3 Input, Output and Constraints

Each of our algorithms takes as input the citizen agent locations $\mathcal{P}$, agent sensing range r and agent radius r_{min}, and outputs mole agent locations $\mathcal{Q}$. They all insert moles incrementally one at a time. At any step, if the previously inserted moles is $\mathcal{Q}$, a potential new mole $\mathbf{q}$ must satisfy the following feasibility constraints. The first constraint prevents physical interference between agents and the second guarantees swarm connectivity.

$$\forall \mathbf{u} \in (\mathcal{P} \cup \mathcal{Q}) : \|\mathbf{q} - \mathbf{u}\|_2 \geq 2r_{min} \tag{4}$$

$$\exists \mathbf{u} \in (\mathcal{P} \cup \mathcal{Q}) : \|\mathbf{q} - \mathbf{u}\|_2 \leq r \tag{5}$$

4.2 Random Scatter Algorithm

Algorithm 1 Random Scatter Algorithm

1: **procedure** ScatterInsertion($\mathcal{P}, r, r_{min}$)
2:     $\mathcal{Q} \leftarrow \varnothing$
3:     **while** $\mathcal{B}_{\mathcal{Q}}(\mathcal{P}) \neq \varnothing$ **do**
4:         $\mathbf{q} \sim U\{\mathbf{x} \in \mathbb{R}^2 \mid (\exists \mathbf{y} \in \mathcal{B}_{\mathcal{Q}}(\mathcal{P}) : \|\mathbf{x} - \mathbf{y}\|_2 \leq r) \wedge$
                $(\forall \mathbf{z} \in (\mathcal{P} \cup \mathcal{Q}) : \|\mathbf{x} - \mathbf{z}\|_2 \geq 2r_{min})\}$
5:         $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{\mathbf{q}\}$
6:     **while** $\exists q \in \mathcal{Q} : \mathcal{B}_{\mathcal{Q} \setminus \{q\}}(\mathcal{P}) = \varnothing$ **do**
7:         $\mathcal{Q} \leftarrow \mathcal{Q} \setminus \{q\}$
8:     **return** $\mathcal{Q}$

Algorithm 1 has two stages: (a) randomly insert moles one at a time at valid insertion locations until no citizens are boundary agents (lines 3–5) and (b) remove unnecessary moles one at a time until no more moles can be removed without making a citizen a boundary agent (lines 6–7). Since this algorithm randomly samples from all valid insertion locations during the first stage, the algorithm will find a solution (i.e. the algorithm is probabilistically complete).

4.3 Grid Search Algorithm

We develop a protrusion-based parameterization for the possible insertion locations of moles to flip citizens in Section 4.3.1 and present grid-search based algorithms in Sections 4.3.2 and 4.3.3.

4.3.1 Mole Agent Insertion

Theorem 2. *When the swarm is connected and contains more than one citizen, only two moles are required to flip a protrusion angle of any boundary citizen agent.*

Proof. In a connected network with more than one agent, every agent is within sensing of at least one other agent. This means that the maximum protrusion angle occurs when an agent $\mathbf{p}_1$ is within sensing range of only one other agent and is on the edge of the other's sensing disk (i.e. $\angle prot_1 = \frac{4\pi}{3}$). From Lemma 1 and arguments in Theorem 1, it is clear that inserting a mole $\mathbf{q}_1$ at $2r_{min}$ from $\mathbf{p}_1$ can result in a $\angle prot_1 \leq \frac{2\pi}{3}$, which allows the citizen to then be flipped by just one more mole $\mathbf{q}_2$. $\square$

Now, having established each boundary agent's protrusion angles are either one-flippable or two-flippable, we examine each case individually. Assume the swarm is composed of multiple citizens.

One-Flippable Protrusion Angle: For boundary agent $\mathbf{p}_i$, we call its protrusion angle $\angle prot_k$ *one-flippable* if it satisfies Equation (3). We define a point on the edge of $\mathbf{p}_i$'s sensing disk as *visible* from another point on the edge if the line segment joining the two does not intersect the disk with radius r_{min}.

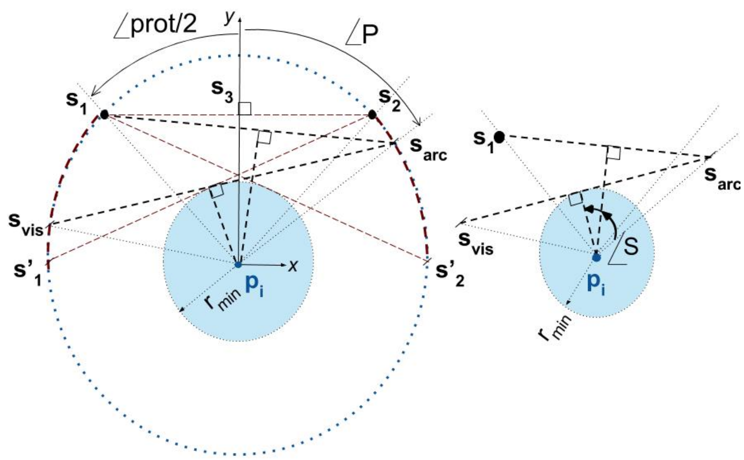


Figure 5: One-flippable citizen: Parameterization of possible mole insertion locations

In Figure 5, $\angle s_1\mathbf{p}_i s_2 = \angle prot$. Segments $\overline{s_2 s_1'}$ and $\overline{s_1 s_2'}$ are tangents to the disk occupied by $\mathbf{p}_i$. Each point s_{arc} on arcs $\widehat{s_1 s_1'}$ and $\widehat{s_2 s_2'}$ has corresponding visible points on the other arc. Each s_{arc} has a corresponding arc $\widehat{s_{vis} s_1}$ which represents its visible points. Each such pair of visible points defines a line segment that would be part of a Voronoi edge within $\mathbf{p}_i$'s sensing disk if a mole was appropriately inserted. Let $\angle S$ represent half the angle such a segment subtends at $\mathbf{p}_i$. Let $\angle P$ be the angle $\overline{\mathbf{p}_i s_{arc}}$ forms with the $y-$axis. Observe, $\angle P \in [\angle s_3\mathbf{p}_i s_2, \angle s_3\mathbf{p}_i s_2']$. For a particular $\angle P$, from Figure 5 (right), $\angle S \in \frac{1}{2}[\angle s_1\mathbf{p}_i s_{arc}, \angle s_{vis}\mathbf{p}_i s_{arc}]$. Then we have:

$$\angle P \in \left[\frac{\angle prot}{2}, 2\cos^{-1}\left(\frac{r_{min}}{r}\right) - \frac{\angle prot}{2}\right] \tag{6}$$

$$\angle S \in \frac{1}{2}\left[\angle P + \frac{\angle prot}{2}, 2\cos^{-1}\left(\frac{r_{min}}{r}\right)\right] \tag{7}$$

Note that the parameterization described above is symmetric about the y-axis. The inserted mole location $(x_{ins},\ y_{ins})$ should be along the perpendicular from $\mathbf{p}_i$ onto the corresponding new Voronoi edge to flip the protrusion angle, so:

$$x_{ins} = \pm 2r\cos(\angle S)\sin(\angle P - \angle S)$$
$$y_{ins} = 2r\cos(\angle S)\cos(\angle P - \angle S)$$

Two-Flippable Protrusion Angle: From Equation (3) and Theorem 2, if protrusion angle $\angle prot_k > 2\cos^{-1}\left(\frac{r_{min}}{r}\right)$, then it requires two moles to flip it and we call $\angle prot_k$ *two-flippable*. In Figure 6, $\angle prot_1 = \angle s_1\mathbf{p}_i s_2$. To flip $\angle prot_1$ with two moles, the protrusion angle must first be made one-flippable (i.e. we must find a mole insertion point such that the *resulting* protrusion angle is less than $2\cos^{-1}\left(\frac{r_{min}}{r}\right)$). The *resulting* pair of protrusion points s_3 and s_2 must, at worst, be as shown. To ensure that this is the resulting protrusion point pair, we treat $\{s_1, s_3\}$ as *secondary protrusion* points, with $\angle prot_{sec} = \angle s_1\mathbf{p}_i s_3$ as the *secondary protrusion angle* (which will always be less than $\angle s_2\mathbf{p}_i s_3 = 2\cos^{-1}\left(\frac{r_{min}}{r}\right)$). Considering this secondary protrusion angle, we insert a mole as we would in the one-flippable case. This ensures that the resulting protrusion angle is one-flippable and is subsequently treated as such. This secondary protrusion is symmetric about the y-axis and thus two such pairs of secondary protrusions exist.

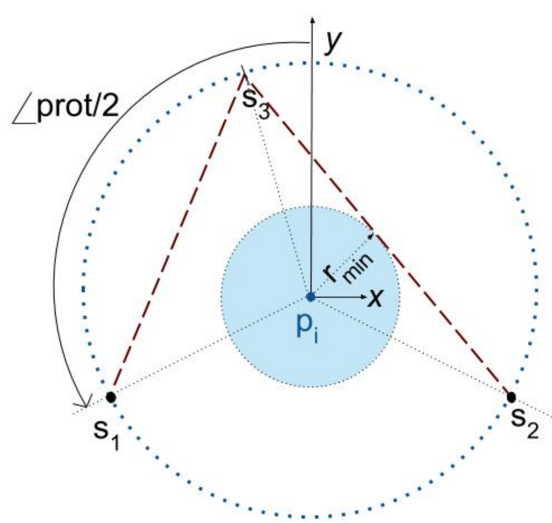


Figure 6: Two-flippable citizen agent: Usage of $\{s_1, s_3\}$ as secondary protrusion points and $\angle s_1\mathbf{p}_i s_3$ as a secondary protrusion angle.

4.3.2 Protrusions Grid Search Insertion

Algorithm 2 does the following: (a) collects protrusion angles of all boundary citizens (lines 5–6), (b) generates grid of potential mole insertion locations with discretization $d = \frac{2\pi}{180}$ in each parameter $\angle P, \angle S$ for each collected angle (line 7), (c) ranks locations, with rank proportional to total sum of protrusion and secondary protrusion angles flipped simultaneously (line 8), (d) adds mole at one of the locations with highest rank

Algorithm 2 Protrusions Grid Search Algorithm

```
1: procedure PROTGSINSERTION(𝒫, r, r_min)
2:     𝒬 ← ∅, d ← 2π/180
3:     while ℬ_𝒬(𝒫) ≠ ∅ do
4:         𝒢 ← {}, 𝒮_all ← {}
5:         for all p ∈ ℬ_𝒬(𝒫) do
6:             𝒮_all ← 𝒮_all ∪ PROTS(p, 𝒱_p(𝒫 ∪ 𝒬), r)
7:         𝒢 ← 𝒢 ∪ GETGRID(ℬ_𝒬(𝒫), 𝒮_all, r, r_min, d)
8:         𝒢_rank ← RANKGRID(𝒢, 𝒮_all, r, r_min)
9:         𝒬 ← 𝒬 ∪ {RAND(BEST(𝒢, 𝒢_rank))}
10:    return 𝒬
```

(line 9), (e) repeats steps (a)–(d) iteratively until there are no more boundary citizens. Since this algorithm considers all possible insertion locations to flip each protrusion angle, it is resolution complete (i.e. it is complete for the chosen level of discretization d).

4.3.3 Randomized Grid Search Insertion

Since Algorithm 2 evaluates all possible insertion locations for each protrusion angle, it takes significantly more time than Algorithm 1 to execute. Since the parameterization developed in Section 4.3.1 is applicable to any protrusion angle, including total protrusion angle $\angle prot_{tot}$, we now attempt to improve the execution time of Algorithm 2 by modifying it to use only the total protrusion angle and randomly choose, at each step, one boundary citizen to flip.

This modification results in Algorithm 3 which (a) identifies boundary citizens $\mathcal{L}_{1flip}$ for which $\angle prot_{tot}$ is one-flippable (line 4), (b) chooses a random citizen from $\mathcal{L}_{1flip}$, generates its parameterized grid of potential mole insertion locations, and inserts a mole according to highest number of agent conversions until there are no more one-flippable agents (line 5), (c) identifies boundary citizens $\mathcal{L}_{2flip}$ for which $\angle prot_{tot}$ is two-flippable (line 6), (d) chooses a random citizen from $\mathcal{L}_{2flip}$, generates its parameterized grid of potential mole insertion locations, and inserts a mole according to highest number of agent conversions (line 7), and (e) repeats (a)–(d) until there are no more boundary citizens. Here, "agent conversions" are from one-flippable to internal or two- to one-flippable.

However, this approach is not complete. There are some situations (e.g. Figure 7) where considering the total protrusion $\angle prot_{tot}$ rather than each protrusion angle $\angle prot_k$ individually results in interference with an existing agent at the potential mole insertion location. In these situations, to enable the algorithm to proceed, the mole is inserted at a random valid location (lines 18–19) similar to Algorithm 1.

Algorithm 3 Randomized Grid Search Algorithm

procedure RANDGSINSERTION($\mathcal{P}, r, r_{min}$)
  $\mathcal{Q} \leftarrow \varnothing, d \leftarrow \frac{2\pi}{180}$
  while $\mathcal{B}_{\mathcal{Q}}(\mathcal{P}) \neq \varnothing$ **do**
    $\mathcal{L}_{1flip} \leftarrow$ GETFLIP($\mathcal{P}, \mathcal{Q}, r, r_{min}$)
    $\mathcal{Q} \leftarrow$ FLIPL($\mathcal{P}, \mathcal{Q}, r, r_{min}, d, \mathcal{L}_{1flip}$)
    $\mathcal{L}_{2flip} \leftarrow \mathcal{B}_{\mathcal{Q}}(\mathcal{P})$
    $\mathcal{Q} \leftarrow$ FLIPL($\mathcal{P}, \mathcal{Q}, r, r_{min}, d, \mathcal{L}_{2flip}$)
  return $\mathcal{Q}$
procedure FLIPL($\mathcal{P},\ \mathcal{Q},\ r,\ r_{min},\ d,\ \mathcal{L}$)
  while $\mathcal{L} \neq \varnothing$ **do**
    $\mathbf{p} \leftarrow$ RAND($\mathcal{L}$)
    $\mathcal{S} \leftarrow$ TOTPROT($\mathbf{p}, \mathcal{V}_{\mathbf{p}}(\mathcal{P} \cup \mathcal{Q}), r$)
    $\mathcal{G} \leftarrow$ GETGRID($\mathbf{p},\ \mathcal{S},\ r,\ r_{min},\ d$)
    $\mathcal{G}_{rank} \leftarrow$ RANKGRID($\mathcal{G},\ \mathcal{S},\ r,\ r_{min}$)
    if $\mathcal{G}_{rank} \neq \varnothing$ **then**
      $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{$RAND$\big($BEST$(\mathcal{G}, \mathcal{G}_{rank})\big)\}$
    else
      $\mathbf{q} \sim U\{\mathbf{x} \in \mathbb{R}^2 \mid (\|\mathbf{x} - \mathbf{p}\|_2 \leq r) \wedge$
        $(\forall \mathbf{z} \in (\mathcal{P} \cup \mathcal{Q}) : \|\mathbf{x} - \mathbf{z}\|_2 \geq 2r_{min})\}$
      $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{\mathbf{q}\}$
    $\mathcal{L} \leftarrow$ GETFLIP($\mathcal{P}, \mathcal{Q}, r, r_{min}$)
  return $\mathcal{Q}$

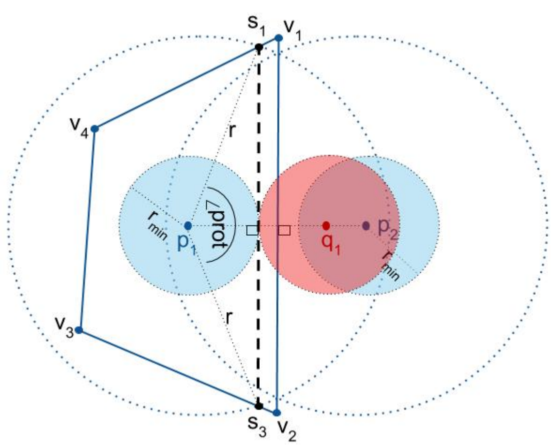


Figure 7: Situation where boundary citizen is incorrectly identified as one-flippable by Algorithm 3 because it considers total protrusion angle rather than individual protrusion angles.

Algorithm 4 Discrete Arc Cover Algorithm

procedure DISCACINSERTION($\mathcal{P}, r, r_{min}$)
  $\mathcal{Q} \leftarrow \varnothing, \mathcal{S}_{all} \leftarrow \varnothing, \mathbf{m} \leftarrow \varnothing, d \leftarrow \frac{\pi}{180}$
  $ol \leftarrow r\sin(d), t \leftarrow 1.5r\sin(d)$
  for all $\mathbf{p} \in \mathcal{B}_{\mathcal{Q}}(\mathcal{P})$ **do**
    $\mathcal{S}_{all} \leftarrow \mathcal{S}_{all} \cup \text{PROTS}(\mathbf{p}, \mathcal{V}_{\mathbf{p}}(\mathcal{P} \cup \mathcal{Q}), r)$
  $\mathcal{A} \leftarrow \text{GETARCPTS}(\mathcal{S}_{all}, \mathcal{P}, \mathcal{B}_{\mathcal{Q}}(\mathcal{P}), d, r)$
  $\mathcal{V}_{ins} \leftarrow \text{GETVALIDINS}(\mathcal{P}, r, r_{min})$
  while $\mathcal{B}_{\mathcal{Q}}(\mathcal{P}) \neq \varnothing$ **do**
    $\mathcal{M}_{ins} \leftarrow \{\mathbf{v} \in \mathcal{V}_{ins} \mid \|\mathbf{v} - \mathbf{m}\|_2 \leq r\}$
    $\mathbf{v}_{ins} \leftarrow \arg\max_{\mathbf{v} \in \mathcal{M}_{ins}} |\{\mathbf{a} \in \mathcal{A} \mid \|\mathbf{v} - \mathbf{a}\|_2 \leq r\}|$
    $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{\mathbf{v}_{ins}\}$
    $\mathcal{A}_c \leftarrow \{\mathbf{a} \in \mathcal{A} \mid \|\mathbf{v}_{ins} - \mathbf{a}\|_2 \leq r\}$
    $\mathcal{O}_c \leftarrow \{\mathbf{a}_c \in \mathcal{A}_c \mid (\ \|\mathbf{v}_{ins} - \mathbf{a}_c\|_2 \geq r - ol\ ) \wedge (\min_{\mathbf{a} \in \mathcal{A}} \|\mathbf{a} - \mathbf{a}_c\|_2 \leq t) \}$
    $\mathbf{m} \leftarrow \arg\min_{\mathbf{o} \in \mathcal{O}_c} \left(\min_{\mathbf{a} \in \mathcal{A} \setminus \mathcal{A}_c} \|\mathbf{o} - \mathbf{a}\|_2 \right)$
    $\mathcal{A} \leftarrow \mathcal{A} \setminus \mathcal{A}_c \cup \mathcal{O}_c$
    $\mathcal{V}_{ins} \leftarrow \{\mathbf{v} \in \mathcal{V}_{ins} \mid \|\mathbf{v} - \mathbf{v}_{ins}\|_2 \geq 2r_{min}\}$
  return $\mathcal{Q}$

Algorithm 5 Greedy Discrete Arc Cover Algorithm

procedure GREEDYDACINSERTION($\mathcal{P}, \mathcal{Q}, r, r_{min}$)
  perform DISCACINSERTION($\mathcal{P}, \mathcal{Q}, r, r_{min}$) with no mandatory overlap point, i.e., $\mathbf{m} \leftarrow \varnothing$ in every incremental insertion

4.4 Discrete Arc Cover Algorithm

4.4.1 Algorithm Description

Previous work in [10] proposes boundary node detection in a sensor network based on finding parts of the perimeter of nodes' sensing disks which are not covered by sensing disks of other nodes. It is straightforward to see that, with uniform sensing ranges across nodes, such 'exposed' arcs form the boundary of coverage holes.

This algorithm places moles such that 'exposed' arcs along the perimeter of boundary agents are covered, so that boundary agents become internal agents. In Figure 3, arcs $\widehat{s_1 s_2}$ and $\widehat{s_3 s_4}$ (i.e. the arcs corresponding to protrusion angles $\angle prot_k$ for the agent $\mathbf{p}_1$) are the exposed parts of the sensing disk $sdisk(\mathbf{p}_1)$ which must be covered by sensing disks of moles to make $\mathbf{p}_1$ an internal agent. Our algorithm discretizes the exposed arcs and the valid domain of insertion satisfying feasibility conditions (4) and (5) and searches over this domain to maximize the number of such 'arc points' covered. However, rather than a simple greedy approach, we also designate a mandatory point of overlap between consecutively inserted mole agents' sensing disks to both (a) minimize redundancy in covering of arc points and (b) ensure complete coverage of exposed arcs despite discretization.

Algorithm 4 does the following: (a) collects protrusion angles of all boundary agents and corresponding arc points $\mathcal{A}$ (lines 4-6), (b) computes set of valid mole insertion locations $\mathcal{V}_{ins}$ (line 7), (c) inserts mole at the best insertion location (lines 9-11), (d) computes overlap arc points $\mathcal{O}$ (lines 12-13), (e) computes next mandatory arc point (line 14), (f) updates $\mathcal{A}$ and $\mathcal{V}_{ins}$ and (lines 15-16) (g) repeats steps (c)-(f) until there are no more boundary citizens.

4.4.2 Greedy version

Algorithm 5 is the greedy version of Algorithm 4 where, unlike Algorithm 4, a mandatory overlap point is not considered in order to ensure minimum redundancy. Instead this performs DISCACINSERTION() such that line 9 results in considering $\mathcal{M}_{ins} = \mathcal{V}_{ins}$ in every incremental insertion — a simple greedy approach.

4.4.3 Bounds on Sub-optimality

Algorithm 5 approaches our problem in a manner that provides guaranteed bounds on its sub-optimality. Given the set of uncovered discretized arc points $\mathcal{A}$ when no moles have been inserted, let $\mathcal{V}_{ins}$ be the discretized set of potential mole insertion locations. Then, for any $\mathbf{v} \in \mathcal{V}_{ins}$, the subset of arc points covered by a mole insertion at $\mathbf{v}$ would be $\mathcal{A}_{cov}(\mathbf{v}) = \{\mathbf{a} \in \mathcal{A} \mid \|\mathbf{a} - \mathbf{v}\|_2 \leq r\}$. Each potential mole insertion location has a corresponding subset of $\mathcal{A}$ which would be covered in case of an insertion at that location. Therefore, our problem is a set-covering problem, where given a set of arc points $\mathcal{A}$ and a finite number of subsets $\mathcal{A}_{cov}(\mathbf{v})$ corresponding to each potential mole insertion location $\mathbf{v}$, our goal is to select the minimum subset of mole insertion locations $\mathcal{Q} \subseteq \mathcal{V}_{ins}$ so that $\mathcal{A} = \bigcup_{\mathbf{v} \in \mathcal{Q}} \mathcal{A}_{cov}(\mathbf{v})$. It is a geometric version of the set-covering problem with discrete unit disks, which is NP-hard [6]. Let the optimal solution to this problem be $\mathcal{Q}_{opt}$. A greedy heuristic, as applied in Algorithm 5, instead incrementally chooses the subset with the maximum number of yet-uncovered arc points until all arc points $\mathcal{A}$ are covered, resulting in a corresponding mole agent set $\mathcal{Q}_g \subseteq \mathcal{V}_{ins}$. Note that $\mathcal{A} = \bigcup_{\mathbf{v} \in \mathcal{Q}_{opt}} \mathcal{A}_{cov}(\mathbf{v}) = \bigcup_{\mathbf{v} \in \mathcal{Q}_g} \mathcal{A}_{cov}(\mathbf{v})$. A well-known result for the greedy heuristic in solving set-covering problems is presented in [5], which proves:

$$\frac{|\mathcal{Q}_g|}{|\mathcal{Q}_{opt}|} \leq \ln \left(\max_{\mathbf{v} \in \mathcal{V}_{ins}} |\mathcal{A}_{cov}(\mathbf{v})| \right)$$

Therefore, Algorithm 5 has a bound on the sub-optimality of the number of moles inserted in terms of the maximum size subset of arc points which may be covered by a mole insertion.

While we do not prove bounds on the sub-optimality of Algorithm 4, we note that the greedy heuristic used by Algorithm 5 does not exploit the inherently contiguous structure of arc points on the edge of sensing holes, which Algorithm 4 does, by designating a mandatory overlap point in sensing to space consecutively inserted moles such that redundancy is reduced. Thus, we expect Algorithm 4 to perform much better than Algorithm 5, and indeed, from results presented in the next section, it does.

4.5 Continuous Arc Cover Algorithm

4.5.1 Algorithm Description

We now take the idea of arc cover to the continuous domain. We no longer discretize the search space, nor do we discretize exposed arcs into arc points to cover. However, in order to search the highly non-convex domain (under constraints (4) and (5)), we perform maximization of our continuous objective of total arc length covered from multiple initial guess points and choose the best. In order to minimize redundancy in coverage of the exposed boundary, we further perform the objective maximization subject to the constraint that a subsequently inserted mole's sensing must cover the previous one's sensing disk's point of intersection with the hole boundary (if it exists). The continuous objective $\textsc{AcOverlap}(\mathbf{o}, \mathcal{P}, \mathcal{Q}) = r\ \big(\sum_j \angle prot_{j,\ (\mathcal{P}\cup\mathcal{Q})} - \sum_j \angle prot_{j,\ (\mathcal{P}\cup\{\mathcal{Q}\cup\mathbf{o}\})}\big)$, where $\sum_j \angle prot_{j,\ (\mathcal{P}\cup\mathcal{X})}$ is the sum of all angles of protrusion for citizen swarm $\mathcal{P}$ in the presence of moles at locations $\mathcal{X}$.

Algorithm 6 does the following: (a) samples a number of start points in valid regions (line 3), (b) finds previous mole's intersection point with the boundary which to subsequently overlap (line 5), (c) performs one of two optimizations: either from samples in the neighbourhood of the previously inserted mole (lines 7-9) or from the previously sampled start points (lines 11-13), (d) chooses the one which maximizes the objective, and (e) repeats steps (b)-(c) till no more boundary citizens remain.

Algorithm 6 Continuous Arc Cover Algorithm

1: **procedure** $\textsc{ContACInsertion}(\mathcal{P}, r, r_{min})$

2: $\quad \mathcal{Q} \leftarrow \varnothing, \mathcal{O} \leftarrow \varnothing, \mathbf{q}_{prev} \leftarrow \varnothing, k \leftarrow 300$

3: $\quad \mathcal{G} \leftarrow \textsc{Sample}\ \big(k, U\{\mathbf{x} \in \mathbb{R}^2 \mid (\forall \mathbf{y} \in (\mathcal{P}\cup\mathcal{Q}) : \|\mathbf{x}-\mathbf{y}\|_2 \geq 2r_{min}) \wedge (\exists \mathbf{z} \in \mathcal{B}_{\mathcal{Q}}(\mathcal{P}) : \|\mathbf{x}-\mathbf{z}\|_2 \leq r)\}\big)$

4: $\quad$ **while** $\mathcal{B}_{\mathcal{Q}}(\mathcal{P}) \neq \varnothing$ **do**

5: $\quad\quad \mathbf{f} \leftarrow \textsc{ArcInter}(\mathbf{q}_{prev}, \mathcal{P}, \mathcal{Q})$

6: $\quad\quad$ **if** $\exists \mathbf{f}$ **then**

7: $\quad\quad\quad \mathcal{N} \leftarrow \textsc{Sample}\ \big(k, U\{\mathbf{x} \in \mathbb{R}^2 \mid (\forall \mathbf{y} \in (\mathcal{P}\cup\mathcal{Q}) : \|\mathbf{x}-\mathbf{y}\|_2 \geq 2r_{min}) \wedge \big(\exists \mathbf{z} \in \mathcal{B}_{\mathcal{Q}}(\mathcal{P}) : \|\mathbf{q}_{prev}-\mathbf{z}\|_2 \leq 2r \wedge \|\mathbf{x}-\mathbf{z}\|_2 \leq r\big)\}\big)$

8: $\quad\quad\quad$ **for all** $\mathbf{n} \in \mathcal{N}$ **do**

9: $\quad\quad\quad \mathcal{O} \leftarrow \mathcal{O} \cup \{ \underset{\mathbf{o}\,:\,\mathbf{o}_{init}\leftarrow\mathbf{n},\ \|\mathbf{o}-\mathbf{f}\|_2 \leq r}{\arg\max} \textsc{ArcOverlap}(\mathbf{o}, \mathcal{P}, \mathcal{Q}), \quad \text{s.t. (4),(5)}\}$

10: $\quad\quad$ **else**

11: $\quad\quad\quad \mathcal{G} \leftarrow \{\mathbf{g} \in \mathcal{G} \mid \forall \mathbf{q} \in \mathcal{Q} : ||\mathbf{g}-\mathbf{q}||_2 \geq 2r_{min}\}$

12: $\quad\quad\quad$ **for all** $\mathbf{g} \in \mathcal{G}$ **do**

13: $\quad\quad\quad\quad \mathcal{O} \leftarrow \mathcal{O} \cup \{ \underset{\mathbf{o}\,:\,\mathbf{o}_{init}\leftarrow\mathbf{g}}{\arg\max} \textsc{ArcOverlap}(\mathbf{o}, \mathcal{P}, \mathcal{Q}), \quad \text{s.t. (4),(5)}\}$

14: $\quad\quad \mathbf{q}_{prev} \leftarrow \underset{\mathbf{o}\in\mathcal{O}}{\arg\max}\ \textsc{ArcOverlap}(\mathbf{o}, \mathcal{P}, \mathcal{Q})$

15: $\quad\quad \mathcal{Q} \leftarrow \mathcal{Q} \cup \{\mathbf{q}_{prev}\}, \quad \mathcal{O} \leftarrow \varnothing$

16: $\quad$ **return** $\mathcal{Q}$

Algorithm 7 Greedy Continuous Arc Cover Algorithm

procedure GREEDYCACINSERTION($\mathcal{P}, r, r_{min}$)
  $\mathcal{Q} \leftarrow \varnothing, \mathcal{O} \leftarrow \varnothing, \mathbf{q}_{prev} \leftarrow \varnothing, k \leftarrow 300$
  $\mathcal{G} \leftarrow \text{SAMPLE}\big(k, U\{\mathbf{x} \in \mathbb{R}^2 \mid (\forall \mathbf{y} \in (\mathcal{P} \cup \mathcal{Q}) : \|\mathbf{x} - \mathbf{y}\|_2 \geq 2r_{min}) \wedge (\exists \mathbf{z} \in \mathcal{B}_{\mathcal{Q}}(\mathcal{P}) : \|\mathbf{x} - \mathbf{z}\|_2 \leq r)\}\big)$
  while $\mathcal{B}_{\mathcal{Q}}(\mathcal{P}) \neq \varnothing$ **do**
    $\mathcal{G} \leftarrow \{\mathbf{g} \in \mathcal{G} \mid \forall \mathbf{q} \in \mathcal{Q} : ||\mathbf{g} - \mathbf{q}||_2 \geq 2r_{min}\}$
    for all $\mathbf{g} \in \mathcal{G}$ **do**
      $\mathcal{O} \leftarrow \mathcal{O} \cup \{\underset{\mathbf{o}\,:\,\mathbf{o}_{init} \leftarrow \mathbf{g}}{\arg\max}\ \text{ARCOVERLAP}(\mathbf{o}, \mathcal{P}, \mathcal{Q}), \quad \text{s.t. (4),(5)}\}$
    $\mathbf{q}_{prev} \leftarrow \underset{\mathbf{o} \in \mathcal{O}}{\arg\max}\ \text{ARCOVERLAP}(\mathbf{o}, \mathcal{P}, \mathcal{Q})$
    $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{\mathbf{q}_{prev}\}, \quad \mathcal{O} \leftarrow \varnothing$
  return $\mathcal{Q}$

4.5.2 Greedy version

As in sub-section 4.4, the greedy version Algorithm 7 simply does not constraint subsequently inserted moles to cover the previous one's intersection with the hole boundary.

5 Algorithm Results and Discussion

5.1 Characteristics Comparison

	Algorithm	Deterministic	Completeness	Strategy
1	SCATTER	No	Probabilistically Complete	Random Sampling
2	PROTGS	No	Resolution Complete	Protrusion Grid Search
3	RANDGS	No	Not Complete*	Protrusion Grid Search*
4	DISCAC	Yes	Resolution Complete	Discrete AC Search
5	GREEDYDAC	Yes	Resolution Complete	Discrete AC Search
6	CONTAC	No	Probabilistically Complete	Continuous AC Search
7	GREEDYCAC	No	Probabilistically Complete	Continuous AC Search

Table 1: Algorithm Theoretical Properties.
(*): RANDGS will switches to random sampling if stuck

We ran extensive experiments in simulation on a computer with dual Intel Xeon CPUs (E5-2660v3, 10 cores @2.60GHz with Hyper-Threading) and 128GB RAM. We varied the number of citizens ($|\mathcal{P}| \in \{10, 20, ..., 100\}$), the configurations of citizens (10 different configurations for each case of $|\mathcal{P}|$) and the ratio of agent radius to sensing radius $r_{min} : r$ ($r = 30, r_{min} \in \{14, 12, 10, 6, 3, 1\}$). In total, 600 different trials were conducted per algorithm. Tables 1 and 2 present a summary comparing the algorithm properties based on our analysis and empirical results from simulation.

	Algorithm	Speed	Domain	Performance
1	Scatter	Fast	Continuous	Very low
2	ProtGS	Moderate	Discrete	Medium
3	RandGS	Fast	Discrete*	Medium
4	DiscAC	Slow	Discrete	High
5	GreedyDAC	Very slow	Discrete	Low
6	ContAC	Moderate	Continuous	High
7	GreedyCAC	Moderate	Continuous	Low

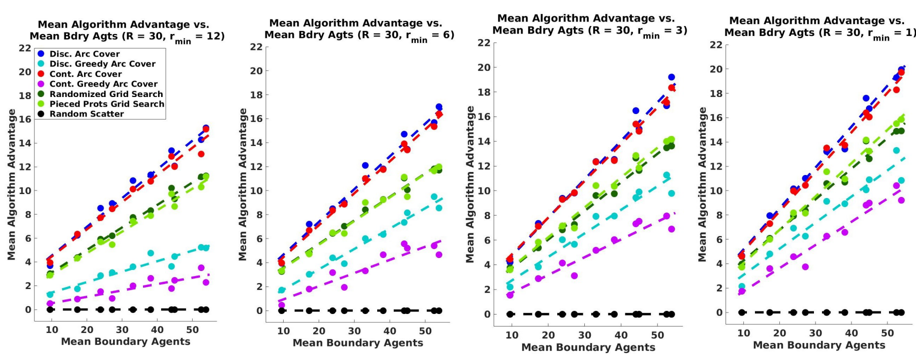
Table 2: Algorithm Experimental Properties.
Speed: wall-clock time. Performance: no. of moles inserted

Algorithm 1 was used as a baseline against which the performance of other algorithms was evaluated. Since some of the algorithms were non-deterministic, they were executed 10 times per trial and the result was averaged across the 10 runs.

In any trial, let moles inserted by Algorithm 1 be $\mathcal{Q}_{scat}$ and for Algorithms 2, 3, 4, 5, 6, 7 be $\mathcal{Q}_{(algo)} \in \{\mathcal{Q}_{PGS}, \mathcal{Q}_{RGS}, \mathcal{Q}_{DCI}, \mathcal{Q}_{DCI_g}, \mathcal{Q}_{CCI}, \mathcal{Q}_{CCI_g}\}$ respectively. We define the 'algorithm advantage' of Algorithms 2, 3, 4, 5, 6, 7 as the corresponding difference: $\big(|\mathcal{Q}_{(algo)}| - |\mathcal{Q}_{scat}|\big)$.

For each of the trials (here we show for $r = 30, r_{min} \in \{12, 6, 3, 1\}$): (i) The average algorithm advantage is plotted against the average number of boundary agents in Figure 8 and (ii) The log ratio of average run time of each algorithm to Algorithm 1 against average number of boundary agents is displayed in Figure 9.

5.2 Algorithm Advantage

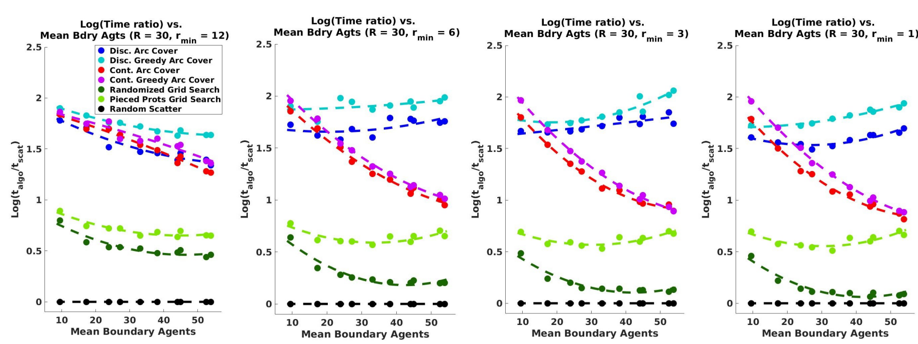


(a) $r_{min} = 12$ (b) $r_{min} = 6$ (c) $r_{min} = 3$ (d) $r_{min} = 1$

Figure 8: Mean advantage of Algorithms over Scatter vs. mean number of boundary agents for $r = 30, \ \ r_{min} \in \{12,\ 6,\ 3,\ 1\}$.

From results in Figure 8 and as noted in Table 2, performance in terms of algorithm advantage over Algorithm 1: GREEDYCAC < GREEDYDAC < PROTGS ≈ RANDGS < DISCAC ≈ CONTAC and this trend holds for all values of $r_{min} : r$ considered. Interestingly, Algorithm 2 and 3 both seem to perform better than greedy Algorithm 5, for which we have proven bounded suboptimality. This is due to the preference for simultaneously flipping as many protrusion angles as possible with a single insertion and a degree of implicit redundancy prevention built into the algorithms. Although Algorithm 2 is more informed at each iteration than 3 since it considers all citizen agents and a total sum of protrusion angles flipped (whereas Algorithm 3 simply picks a random citizen agent's $\angle prot_{tot}$ and considers total number of agents flipped), the two have nearly the same performance. The greedy version of the continuous-domain Algorithm 7 performs worse than the discrete-domain Algorithm 5 since it performs very exact computations of insertion locations leading to more small 'gaps' left in boundary coverage, subsequently leading to higher redundancy. Algorithms 4 and 6 are by far the best performer in every case with an average algorithm advantage of ≈ 21 mole agents for $|\mathcal{B}_{\varnothing}(\mathcal{P})| \approx 54$, when $r_{min} = 1$. Their dominance is expected, as both Algorithms 2 and 3 have limited implicit handling of redundancy in arc coverage. We observe the general decrease in advantage across all algorithms in going from $r_{min} = 14$ to 10, and a gradual increase from $r_{min} = 6$ to 1, indicating an advantage 'valley' between the $r_{min} : r$ ratios of 1:3 and 1:5.

5.3 Run-time Ratio



(a) $r_{min} = 12$ (b) $r_{min} = 6$ (c) $r_{min} = 3$ (d) $r_{min} = 1$

Figure 9: Log mean run time ratio of Algorithms to SCATTER vs. mean number of boundary robots for $r = 30, \ \ r_{min} \in \{12, \ 6, \ 3, \ 1\}$.

From results in Figure 9 and as noted in Table 2, general performance in terms of run time ratio against Algorithm 1: GREEDYDAC < DISCAC < GREEDYCAC < CONTAC < PROTGS < RANDGS. For higher $r_{min} : r$ ratios, Algorithms 4, 5, 6, 7 perform similarly. Algorithm 1 runs in 0.05-1.5 seconds for $r = 30, r_{min} \in \{14, 10\}$ and in 0.1-2.3 seconds for $r = 30, r_{min} \in \{6, 3, 1\}$ with increasing $|\mathcal{B}_{\varnothing}(\mathcal{P})|$. Interestingly, Algorithm 3 is fastest after Algorithm 1 due to randomization, with the run-time log ratio falling almost to 0 as r_{min} decreases. Algorithm 2 comes in next, with nearly constant ratios across r_{min} values. In fact, with large $r_{min} : r$ ratio and low $|\mathcal{B}_{\varnothing}(\mathcal{P})|$, Algorithm 2 and Algorithm 3 perform similarly. Algorithm 4 is an order of magnitude slower. However, even for the most computationally expensive case of $r_{min} = 1, |\mathcal{B}_{\varnothing}(\mathcal{P})| \approx 54$, its run time was under 2 minutes. Greedy Algorithm 5 is slowest, as this considers every remaining valid point of insertion in each iteration. By contrast, Algorithm 4 only considers those insertion locations which are within r of its mandatory overlap point in each iteration. Run times for the continuous-domain Algorithms 6 and 7 are nearly invariable with number of boundary citizens, and thus they very quickly approach the fast speeds of Algorithms 2 and 3, especially at higher number of boundary citizens.

5.4 Overall Observations

It is surprisingly apparent among Algorithms 2, 3, 4, 5, 6 that bounded sub-optimal Algorithm 5 is the worst empirical performer under both measures. Algorithms 4, 6 afford the best algorithm advantage in terms of number of moles required, whereas Algorithm 3 is an order of magnitude faster, running in only a few seconds in every case. Interestingly, among the protrusions-based grid search algorithms, despite the fact that Algorithm 3 has no completeness guarantees, it is generally faster than Algorithm 2 and inserts a similar number of moles, indicating that randomization plays a beneficial role in improving speed without negatively affecting performance (in terms of algorithm advantage). In fact, it is only similar to Algorithm 2 in run time at large $r_{min} : r$ ratios and low $|\mathcal{B}_{\varnothing}(\mathcal{P})|$, where randomization does not provide as much advantage. Algorithm 6 is able to achieve similar best performance using higher speeds than Algorithm 4; in fact, its speeds quickly approach that of 2. Admittedly, Algorithm 7 has low utility as it is even empirically worse in performance than bounded-suboptimal Algorithm 5.

We now discuss different scenarios of mole insertion. In our formulation, all algorithms operate on static snapshots of the citizen agent swarm. However, as mentioned in Sections 5.3 and 5.4 and visible by comparison in the plots, Algorithms 1, 2, and 3 have a run-time within hundreds of milliseconds - this would enable us to recompute and update insertion locations in near-real-time for moving goals on the fly in dynamic multi-agent systems. All algorithms are also applicable to static sensor networks, with Algorithms 4, 6 being most applicable in such cases and in cases of high cost per mole, since performance would then be more important than run-time. Furthermore, for citizen swarms operating in a 2D plane in a 3D world, the third dimension may be used for mole insertion. For example, in an aerial swarm or a swarm operating on the ocean surface, moles may be inserted from a different elevation or submerged moles may rise to the surface.

6 Dynamic Problem

We have so far considered a given set of citizens $\mathcal{P}$ and tried to identify a minimum size mole set $\mathcal{Q}$ so that its insertion would result in $\mathcal{B}_{\mathcal{Q}}(\mathcal{P}) = \varnothing$ under robot and sensing radius constraints. We had assumed that as many agents as would be required would be available at our disposal to insert into these locations. We have described situations in which knowing these destination locations would be enough to insert moles and prevent hole detection.

We now consider a different scenario: suppose we are given a limited set of moles at certain initial locations in our plane. Our goal is now to identify control laws to move the moles so as to minimize the final length of the 'exposed' boundary arcs of the citizens $\mathcal{P}$. In Figure 10, we see an example configuration of citizens in blue, and ten moles in red scattered about the plane, before, while and after applying control to minimize exposed coverage hole boundaries.

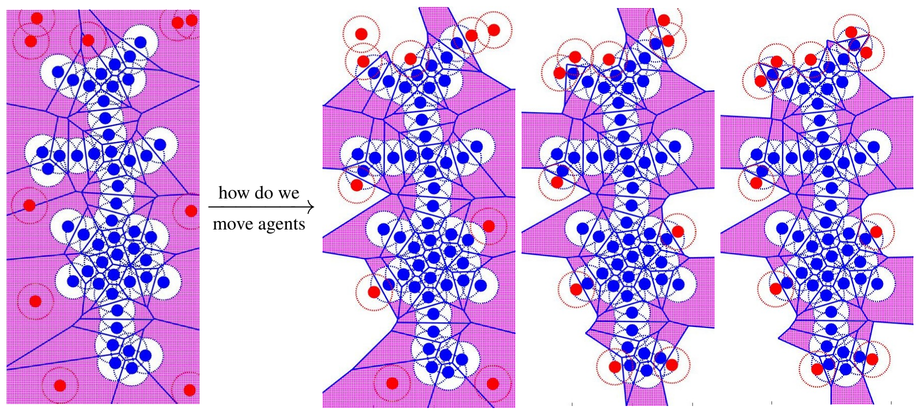


Figure 10: Moving moles in order to minimize exposed coverage hole boundary

Formally, we wish to move the agents (say m agents) given into final positions $\mathcal{Q}$ which minimize the following objective (subject to the same interference and connectivity constraints):

$$\begin{array}{ll} \arg\min\limits_{\mathcal{Q}} & r\sum\limits_{j} \angle prot_{j,\,(\mathcal{P}\cup\mathcal{Q})} \\ \text{subject to} & |\mathcal{Q}| = m \\ \forall \mathbf{u}_k, \mathbf{u}_l \in (\mathcal{P}\cup\mathcal{Q}) & : \ \|\mathbf{u}_k - \mathbf{u}_l\|_2 \geq 2r_{min} \\ \lambda\left(L(G)\right) = \left\{\lambda_1, \lambda_2, \ldots, \lambda_{|\mathcal{P}\cup\mathcal{Q}|}\right\} & : \lambda_2 > 0 \end{array}$$

From sub-section 4.5 we have a continuous objective function in the length of boundary arcs covered, whose gradient gives us a natural way of moving moles to minimize hole boundaries, i.e., in the direction of maximum ascent. Let this covering function (called ArcOverlap() previously) be $f_{arc}(\mathbf{x}, \mathcal{P}, \mathcal{Q})$ at the plane position $\mathbf{x} \in \mathbb{R}^2$, with current citizen positions $\mathcal{P}$ and mole positions $\mathcal{Q}$.

In Figure 11, we see an example citizen configuration with the objective value of the function $f_{arc}()$ displayed, warmer colors indicating higher objective values. Also displayed using black arrows are the gradient directions (the size of the arrow indicating magnitude). Zoomed-in views are also shown in this figure.

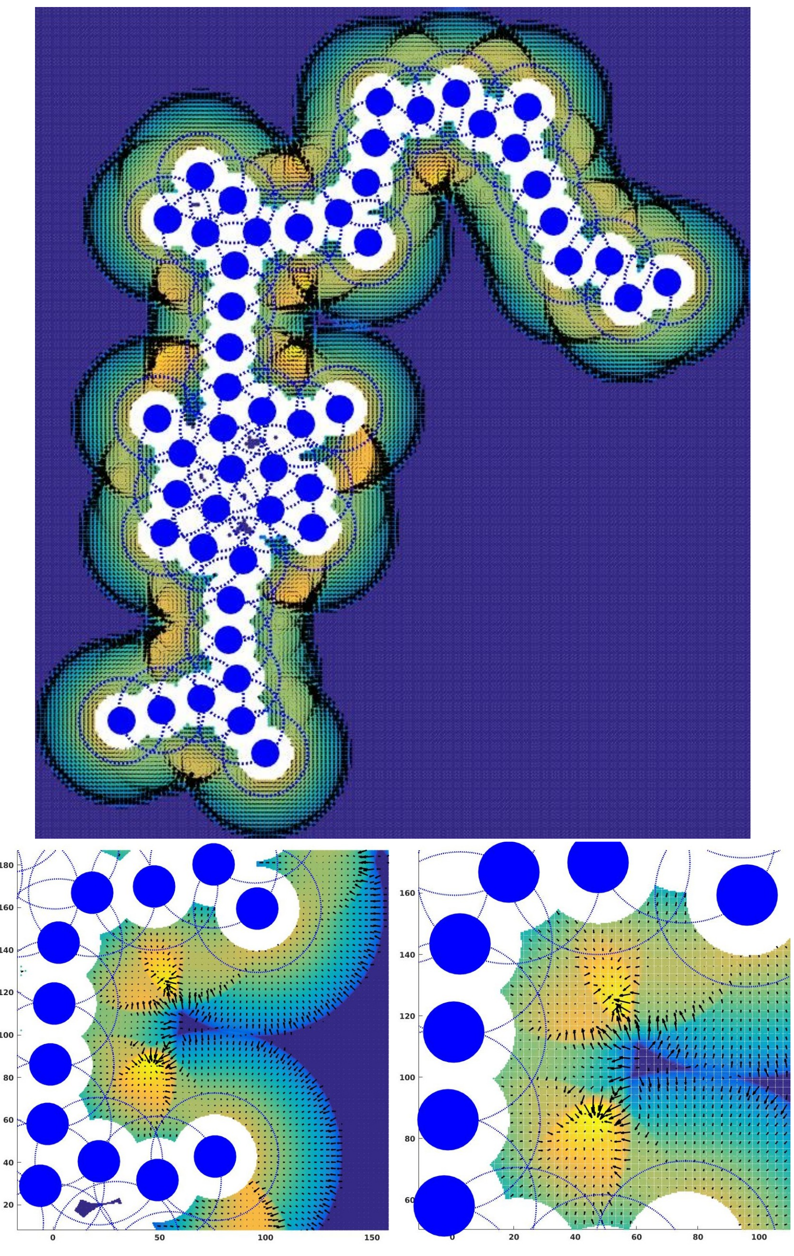


Figure 11: 'Exposed' boundary coverage: objective displayed for example citizen configuration; warmer colour denotes higher function value. Black arrows represent magnitude and direction of gradient; zoomed in views are shown.

We now describe control laws in order to move our moles. We will see that, since every law is purely reactive and does not depend on projected citizen locations, they may be used without alteration both in the presence and absence of citizen motion. Each mole $\mathbf{q}^i \in \mathcal{Q}$ has control inputs such that $\dot{\mathbf{q}}^i = \mathbf{u}^i_{rep} + \mathbf{u}^i_c$.

Each mole is repelled away from any other moles or citizens in a neighborhood within a radius of repulsion r_{rep}, i.e., $\mathcal{N}^i = \{\mathbf{x} \in (\mathcal{P} \cup \mathcal{Q}) \mid \left\|\mathbf{q}^i - \mathbf{x}\right\|_2 \leq r_{rep}\}$, so that for scalar k_{rep}, we have $\mathbf{u}^i_{rep} = k_{rep} \sum\limits_{\mathbf{x} \in \mathcal{N}^i} \frac{\mathbf{q}^i - \mathbf{x}}{\|\mathbf{q}^i - \mathbf{x}\|_2}$.

Each mole has an additional control input $\mathbf{u}^i_c$, according to which zone away from the citizens $\mathcal{P}$ they are in. Let nearest citizen distance $d_{near} = \min\limits_{\mathbf{p} \in \mathcal{P}} \left\|\mathbf{q}^i - \mathbf{p}\right\|_2$. We use this quantity and the agents' sensing range r to identify three possible zones:

1. If $d_{near} > 2r$:

   In this zone, none of the exposed boundary can be within sensing of this mole, and a gradient is absent. We want to move moles towards citizens. Let $\mathbf{p}^i$ be a citizen uniquely assigned to each mole $\mathbf{q}^i$ in order of euclidean proximity. Then in this zone, for scalar k_{far}, we have:

   $$\mathbf{u}^i_c = k_{far} \frac{\mathbf{p}^i - \mathbf{q}^i}{\|\mathbf{p}^i - \mathbf{q}^i\|_2}$$

2. If $2r \geq d_{near} > r$:

   We consider the gradient $\mathbf{g}^i = \nabla_{\mathbf{x}} f_{arc}(\mathbf{q}^i,\ \mathcal{P},\ \mathcal{Q} \setminus \mathbf{q}^i)$. We define a scalar threshold for the gradient norm t_g. If the gradient norm is not negligible, we naturally move the mole in the direction of maximum ascent.

   If the norm is small, this occurs due to one of two reasons: (a) the mole has reached a local maxima or (b) the gradient is absent as any part of the exposed boundary being covered by the mole is already covered by other moles. Neither situation is useful, as, in (a), the mole cannot be sensed by the citizens and will thus have no effect on their localized Voronoi computation. In (b), it is obvious that the mole is redundant. In such a situation, we will do one of two things: (i) move inward toward a citizen if the projection of the mole onto its sensing disk causes a non-zero coverage of exposed boundaries, or (ii) move along the 'wall' formed by the hole boundary, i.e., move perpendicular to the nearest citizen direction in search of exposed boundaries elsewhere. An example of directions the mole would move in under the two conditions is depicted in Figure 12.

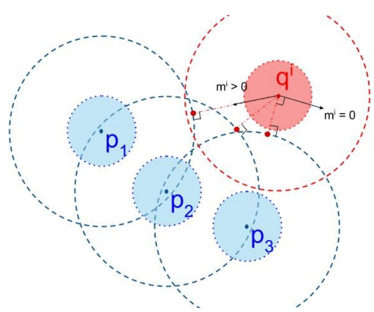


Figure 12: Zone 2 behavior of mole in dynamic problem

Let $P_C(\mathbf{x})$ be the projection of vector $\mathbf{x}$ onto the set C. For scalars k_g, k_s, we have:

$$\mathbf{u}_c^i = \begin{cases} k_g \mathbf{g}^i, & \text{if } \left\|\mathbf{g}^i\right\|_2 > t_g \\ k_s \frac{\mathbf{p}^{m,i} - \mathbf{q}^i}{\|\mathbf{p}^{m,i} - \mathbf{q}^i\|_2}, & \text{if } \left\|\mathbf{g}^i\right\|_2 \leq t_g, \;\; m^i > 0 \\ k_s \left(\frac{\mathbf{p}^{n,i} - \mathbf{q}^i}{\|\mathbf{p}^{n,i} - \mathbf{q}^i\|_2} \right)^{\perp}, & \text{if } \left\|\mathbf{g}^i\right\|_2 \leq t_g, \;\; m^i = 0 \end{cases}$$

where

$$\begin{aligned} \mathbf{p}^{m,i} &= \underset{\{\mathbf{p} \in \mathcal{P} : \|\mathbf{q}^i - \mathbf{p}\|_2 \leq 2r\}}{\arg\max} f_{arc}\big(P_{\{\mathbf{x} : \|\mathbf{x} - \mathbf{p}\|_2 \leq r\}}(\mathbf{q}^i),\ \mathcal{P},\ \mathcal{Q} \setminus \mathbf{q}^i\big) \\ m^i &= f_{arc}\big(P_{\{\mathbf{x} : \|\mathbf{x} - \mathbf{p}^{\mathbf{m},\mathbf{i}}\|_2 \leq r\}}(\mathbf{q}^i),\ \mathcal{P},\ \mathcal{Q} \setminus \mathbf{q}^i\big) \\ \mathbf{p}^{n,i} &= \underset{\mathbf{p} \in \mathcal{P}}{\arg\min} \left\|\mathbf{q}^i - \mathbf{p}\right\|_2 \end{aligned}$$

$$(\mathbf{x})^{\perp} \text{ denotes vector } \mathbf{x} \text{ rotated by } \frac{\pi}{2}$$

3. If $r \geq d_{near}$:

   We consider the gradient $\mathbf{g}^i = \bigtriangledown_{\mathbf{x}} f_{arc}(\mathbf{q}^i,\ \mathcal{P},\ \mathcal{Q} \setminus \mathbf{q}^i)$. We define a scalar threshold for the gradient norm t_g. If the gradient norm is not negligible, we naturally move the mole in the direction of maximum ascent.

   If the norm is small, we simply make sure moles are spaced well apart throughout the exposed boundary by adding repulsion between moles in a certain neighborhood. For scalars k_g, k_{rep}, we have:

$$\mathbf{u}_c^i = \begin{cases} k_g \mathbf{g}^i, & \text{if } \left\|\mathbf{g}^i\right\|_2 > t_g \\ k_{rep} \sum\limits_{\mathbf{x} \in \mathcal{N}^i} \frac{\mathbf{q}^i - \mathbf{x}}{\|\mathbf{q}^i - \mathbf{x}\|_2}, & \text{if } \left\|\mathbf{g}^i\right\|_2 \leq t_g \end{cases}$$

   where

$$\begin{aligned} \mathcal{N}^i &= \{\mathbf{q} \in \mathcal{Q} \setminus \mathbf{q}^i \mid \left\|\mathbf{q} - \mathbf{q}^i\right\|_2 < 2r\cos(\frac{\pi}{6}) \ \wedge \ \left\|n(\mathbf{q}) - n(\mathbf{q}^i)\right\|_2 \leq 2r\} \\ n(\mathbf{x}) &= \underset{\mathbf{p} \in \mathcal{P}}{\arg\min} \|\mathbf{x} - \mathbf{p}\|_2 \end{aligned}$$

Videos: These laws were tested out in cases with $\{10, 30, 50, 70\}$ citizens, with $r_{min} = 10,\ r = 30$ both in the presence and absence of citizen motion. The videos of these trials can be found at this link. Here, we see the citizens in blue, the moles in red, and one mole $\mathbf{q}^i$ in magenta whose objective value $f_{arc}(\mathbf{x}, \mathcal{P}, \mathcal{Q} \setminus \mathbf{q}^i)$ is displayed throughout the plane with warmer colour depicting higher objective value.

7 Conclusions and Future Work

In this research, we explored novel problems for robotic swarms. We identified a swarm vulnerability and studied how an adversary can take advantage of this vulnerability to find the best locations to insert mole agents so as to prevent the original swarm from discovering and repairing faulty performance (in our case the faulty performance consists of leaving holes in area coverage in a surveillance mission). To the best of our knowledge, this is the first to study this problem. We formalized the problem and devised supporting theory and algorithmic solutions. Furthermore, we experimentally evaluated our algorithms, and presented and discussed their different efficiency, performance characteristics and tradeoffs. Finally, based on developed theory and algorithms, we presented control laws governing adversary movement to leverage this vulnerability.

In future work, we plan to (a) identify ways for the swarm to protect itself from such mole insertions, and (b) identify any additional swarm vulnerabilities.

References

[1] Nadeem Ahmed, Salil S Kanhere, and Sanjay Jha. "The holes problem in wireless sensor networks: a survey". In: *ACM SIGMOBILE Mobile Computing and Communications Review* 9.2 (2005), pp. 4–18.

[2] Franz Aurenhammer. "Voronoi diagrams – a survey of a fundamental geometric data structure". In: *ACM Computing Surveys (CSUR)* 23.3 (1991), pp. 345–405.

[3] Manuele Brambilla et al. "Swarm robotics: a review from the swarm engineering perspective". In: *Swarm Intelligence* 7.1 (2013), pp. 1–41.

[4] Xiangqian Chen et al. "Sensor network security: a survey". In: *IEEE Communications Surveys & Tutorials* 11.2 (2009), pp. 52–73.

[5] Vasek Chvatal. "A greedy heuristic for the set-covering problem". In: *Mathematics of operations research* 4.3 (1979), pp. 233–235.

[6] Gautam K Das et al. "On the discrete unit disk cover problem". In: *International Workshop on Algorithms and Computation*. Springer. 2011, pp. 146–157.

[7] Jason Derenick, Vijay Kumar, and Ali Jadbabaie. "Towards simplicial coverage repair for mobile robot teams". In: *Robotics and Automation (ICRA), 2010 IEEE International Conference on*. IEEE. 2010, pp. 5472–5477.

[8] Stefan Funke. "Topological hole detection in wireless sensor networks and its applications". In: *Proceedings of the 2005 joint workshop on Foundations of mobile computing*. ACM. 2005, pp. 44–53.

[9] Robert Ghrist and Abubakr Muhammad. "Coverage and hole-detection in sensor networks via homology". In: *Proceedings of the 4th international symposium on Information processing in sensor networks*. IEEE Press. 2005, p. 34.

[10] Chi-Fu Huang and Yu-Chee Tseng. "The coverage problem in a wireless sensor network". In: *Mobile Networks and Applications* 10.4 (2005), pp. 519–528.

[11] Andreas Kolling et al. "Human Interaction With Robot Swarms: A Survey". In: *IEEE Transactions on Human-Machine Systems* 46.1 (2016), pp. 9–26.

[12] Prasan Kumar Sahoo, Ming-Jer Chiang, and Shih-Lin Wu. "An Efficient Distributed Coverage Hole Detection Protocol for Wireless Sensor Networks". In: *Sensors* 16.3 (2016), p. 386.

[13] Bojan Mohar et al. "The Laplacian spectrum of graphs". In: *Graph theory, combinatorics, and applications* 2.871-898 (1991), p. 12.

[14] Martin EW Nisser et al. "Feedback-controlled self-folding of autonomous robot collectives". In: *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*. IEEE. 2016, pp. 1254–1261.

[15] Juan Rada-Vilela, Mark Johnston, and Mengjie Zhang. "Population statistics for particle swarm optimization: Resampling methods in noisy optimization problems". In: *Swarm and Evolutionary Computation* 17 (2014), pp. 37–59.

[16] Michael Rubenstein, Christian Ahler, and Radhika Nagpal. "Kilobot: A low cost scalable robot system for collective behaviors". In: *Robotics and Automation (ICRA), 2012 IEEE International Conference on*. IEEE. 2012, pp. 3293–3298.

[17] Paul Scharre. “Robotics on the Battlefield Part II: The Coming Swarm”. In: *Center for a New American Security* 6 (2014).

[18] Jaydip Sen. “A Survey on Wireless Sensor Network Security”. In: *International Journal of Communication Networks and Information Security (IJCNIS)* 1.2 (2009).

[19] Alan R Wagner and Ronald C Arkin. “Acting deceptively: Providing robots with the capacity for deception”. In: *International Journal of Social Robotics* 3.1 (2011), pp. 5–26.

[20] Guiling Wang, Guohong Cao, and Thomas F La Porta. “Movement-assisted sensor deployment”. In: *IEEE Transactions on Mobile Computing* 5.6 (2006), pp. 640–652.

[21] Mohamed Younis and Kemal Akkaya. “Strategies and techniques for node placement in wireless sensor networks: A survey”. In: *Ad Hoc Networks* 6.4 (2008), pp. 621–655.

[22] Chi Zhang, Yanchao Zhang, and Yuguang Fang. “Localized algorithms for coverage boundary detection in wireless sensor networks”. In: *Wireless networks* 15.1 (2009), pp. 3–20.