CYBERNETICS AND PHYSICS, VOL. 14, NO. 1, 2025, 52–61

DISTRIBUTED ALGORITHMS FOR SELF-SPREADING OF ROBOTIC NETWORKS OVER UNKNOWN COMPLEX AREAS IN GPS-DENIED ENVIRONMENTS

Petr Konovalov
Mathematical Robotics Science Division
Sirius University
Sirius City, Krasnodar region, Russia
petrkon98@gmail.com

Alexey Matveev
Institute for Problems in Mechanical Engineering
Russian Academy of Sciences
Saint Petersburg, V.O., Bolshoj prosp., 61, Russia
almat1712@yahoo.com

Article history:
Received 29.01.2025, Accepted 21.06.2025

Abstract

This paper presents new algorithms for area coverage by mobile robotic swarms in complex and unknown environments. The robots are silent and not aware of the team's size, do not discern between each other, lack access to a positioning system, and cannot play distinct roles and so should be driven by a common control rule. On the positive side, they determine the relative positions of the objects, including the boundary of the handled area and obstacles, within a given finite sensing range and have access to a common direction. The proposed algorithms are attributed the physics-inspired virtual force-based (VFB) approach. To highlight their benefits, the paper reports on comparative analysis of seven methods, including the above two ones and an essential group of well-established algorithms that follow VFB approach and are suited to handle the examined scenario. The analysis is carried out using a whole range of various metrics, which capture different aspects of the performance quality. Moreover, a new criterion of coverage uniformity is introduced and justified. Extensive computer simulations in complex scenes have shown that the proposed algorithms demonstrate the best performance in terms of coverage uniformity and percentage.

1 Introduction

Rapid advances in robotics, computing, and communications have enabled practical deployment of self-organized networks using inexpensive robotic devices. These devices cooperate through sensing or communication to accomplish common tasks such as monitoring, processing, or ensuring communication connectivity [Savkin et al., 2015,Cortés and Egerstedt, 2017,Wang et al., 2018]. Applications include target detection and tracking; surveillance and rescue operations (e.g., detecting individuals trapped in fires at burning buildings); autonomous deployment of ad-hoc wireless communication or sensor networks; mini-satellite control in low earth orbit [Somov et al., 2023]; precision farming; virtual antenna array construction [Wang et al., 2018]; underwater pollution source identification; and quadcopter synchronization [Sharma and Lather, 2024]. The introduction of 5G and IoT technologies significantly enhances robotic networks' capabilities in coverage missions.

In resource-constrained networks, nodes face limitations in detection range, communication, and operational capabilities. Effective performance requires strategic placement to ensure full area coverage. Due to a range of factors, including poor knowledge on the area, its inaccessibility or dangerousness, a critical option is sensor-based self-distribution of mobile nodes. Algorithmic homogeneity—where nodes execute identical rules without peer differentiation—offers benefits including fault tolerance, scalability, and cost-effectiveness, aligning with current swarm robotics research [Muhsen et al., 2024]. Energy efficiency necessitates halting nodes post-coverage while maintaining self-reorganization capacity for node failures.

Distributed algorithms for self-spreading of homogeneous robotic teams have gained significant attention in recent decades. For detailed overviews and taxonomies, we refer readers to [Sadeghi Ghahroudi et al., 2023, Muhsen et al., 2024]. Briefly, a large category consists of metaheuristic methods inspired by animal aggregation phenomena and swarm intelligence [Houssein et al., 2024, Muhsen et al., 2024]. Representative examples include ant colony optimization, particle swarm optimization, bacterial foraging optimization, and bee algorithms. Other approaches borrow concepts from immunology, genetics, and evolutionary biology. Within

this category, stochastic mechanisms are common, and research objectives often exclude requirements for eventual node stoppage and complete static coverage formation.

Another category comprises physicomimetics-based methods. Dominated by deterministic techniques, these approaches draw significant inspiration from potential force fields and molecular equilibrium phenomena [Sadeghi Ghahroudi et al., 2023, Muhsen et al., 2024]. Although classically understood in terms of forces, many methods are ultimately formulated using velocity or single-step displacement vectors (see e.g., [Ma et al., 2008]). The key feature is that each node deterministically calculates a movement vector based on local information to determine its immediate motion.

A third abundant category is distinguished by its clear-cut reliance on concepts and techniques of geometry [Sadeghi Ghahroudi et al., 2023, Muhsen et al., 2024]. Using a Voronoi tessellation is a popular approach here. Another approach is based on access of all nodes to a common global geometric structure, e.g., a regular grid or cell decomposition. For further discussion of the area, we refer the reader to the specialized surveys, see e.g., [Priyadarshi et al., 2022, Sadeghi Ghahroudi et al., 2023, Muhsen et al., 2024] and the literature therein.

These algorithms pursue diverse objectives, such as filling coverage holes. Our focus is robots starting from a small disembarkation point that must self-distribute to cover large complex areas and ultimately halt. This is particularly critical for GPS/LPS-denied environments lacking communication and prior scene knowledge.

On the one hand, these limitations narrow down the diversity of the relevant algorithms. On the other hand, the remaining group is still ample and ever-growing. This is by itself a sign of a lack of an absolute leader. Also, choice from so many options is no easy task, especially since a rigorous analysis of convergence is typically an intricate matter in this field so that computer simulations and sometimes real-life experiments come to the fore.

Motivated by the foregoing, this paper pursues and offers the following objectives and contributions.

i) We report on the findings from our comparative analysis of an essential group of deterministic algorithms of nodes self-spreading that are well suited to handle the outlined scenario of our interest and can be attributed to the VFB approach. Partly based on the analysis from [Sadeghi Ghahroudi et al., 2023], we select the algorithms **DSSA** [Heo and Varshney, 2003], **SSND** [Ghahroudi et al., 2019], **SWARM** [Mathews et al., 2012], **VFA-SF** [Deng et al., 2019], and **SODA** [Ghahroudi et al., 2018] on the ground that they embody key and somewhat keystone ideas of the group.

In the documents known to the authors, pair-wise and sometimes triple-wise comparisons were addressed. This hampers capturing the whole picture and selecting an algorithm whose individual combination of strengths and weaknesses best fits a particular scenario at hands.

ii) Being motivated by the results from i) and concerns about the distribution uniformity, we offer two new VFB algorithms and include them in the set analysed in i).

According to our study, they overall tend to perform better w.r.t. deployment uniformity in terms of both area coverage and distances travelled and also w.r.t. the coverage percentage. They demonstrate a better coverage quality in complex scenes, enabling the network to cover the entirety of the area in scenarios where the competitors leave considerable subareas uncovered. Also, the rules regulating the phases of the network self-spreading and stoppage, respectively, are identical for the new algorithms. The benefit from this is that if an already built static coverage is violated, the nodes automatically resume moving in order to fix the damage.

iii) For the integrity of comparison, we simultaneously use the most of the performance metrics commonly found in the literature. They capture different aspects of the quality of service: time elapsed until the halt of the network, eventual coverage percentage, energy consumption, mean distance traveled by the nodes, and the uniformity of the distances.

To the best of the knowledge of the authors, comparative analysis was previously performed by using only a few, typically one, criterion. This did not lay ground for a clear picture when selecting a particular method.

Also, we offer, advocate, and use a new and complementary criterion: the *area coverage uniformity*. It assesses the degree of uniformity in the distribution of the network resources over the area. This may be of interest to ensure, e.g., equal tolerance of various parts of the network to faults and so robustness of the overall network to faults in unknown and equiprobable parts.

iv) We test the algorithms in complex environments cluttered with obstacles.

Previously, node deployments were largely tested in convex and simple, so to say, prototypical areas without obstacles. There are not so much algorithms for coverage of complex areas with obstacles; see, e.g., [Bartolini et al., 2017, Eledlebi et al., 2022]). Mostly, they are specially fitted to this case and do not meet all limitations that define the scenario studied in this paper. As is emphasized in, e.g., [Ghahroudi et al., 2022], broad comparison of area coverage methods in complex scenes basically lies in an uncharted territory.

The implicit focus of this article is on missions where a moderate swarm's size of about dozens of robots is enough, like in some missions on precision farming in agriculture [Mahbub, 2020], or underwater surveillance [Ferri et al., 2017], or building virtual antenna arrays [Wang et al., 2018]. As a result, exactly such sizes are considered in our experiments.

For the convenience of the reader, the paper offers pseudocodes of all compared algorithms.

The body of the paper is organised as follows. In Sec. 2, we formulate the problem. Section 3 introduces the proposed algorithms, whereas Section 4 discusses various performance metrics. Section 5 reports on the results

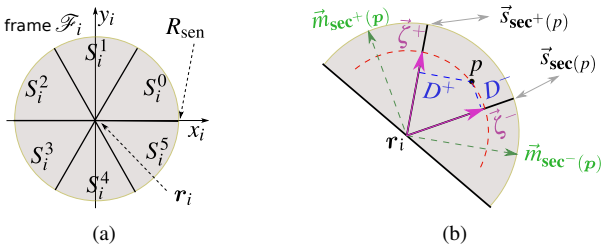


Figure 1. (a) Zone M_i and frame $\mathcal{F}_i$; (b) Building semi-reflections.

of computer simulation tests and based on them, offers a comparative analysis of several algorithms. Section 6 offers brief conclusions and describes future work.

2 Problem statement

A team of N point-wise mobile robots operates in a plane $\mathbb{R}^2$. The position of robot i is denoted by $\boldsymbol{r}_i \in \mathbb{R}^2$. Unless otherwise stated, we assume robots with continuous-time simple integrator kinematics:

$$\dot{\boldsymbol{r}}_i(t) = u_i(t) \qquad \forall t \geq 0, i = 1, \ldots, N. \tag{1}$$

Here $u_i(t)$ is the control input built by robot i at time t.

Any robot i has its own *monitoring zone* M_i: robot i can detect objects from M_i and determine their relative positions. This zone is the disc centered at $\boldsymbol{r}_i$ and with a radius of R_{sen}. The robots are *anonymous* to each other, i.e., they cannot distinguish among the peers.

The team should be spatially distributed in such a way that every point from a certain a priori unknown *region* $\mathcal{R} \subset \mathbb{R}^2$ is covered by the monitoring zone of some robot. More precisely, a motion control rule is to be designed such that when being individually run by every robot, it eventually creates a situation where the just stated objective is attained and all robots halt. Any robot has the capacity to detect the edge of $\mathcal{R}$ within the monitoring zone; the robots are initially in $\mathcal{R}$.

In practice, there are scenarios where robot i somehow influences close objects or communicates with them, and it is needed that eventually any point of $\mathcal{R}$ is affected by or communicates with some robot. The model employed in this paper assumes that the distances of monitoring, influence, and communication (if applicable) are equal; otherwise, they are to be artificially reduced to the least of them prior to the use of the algorithms to be discussed.

The following notations are further adopted:

- $|E|$, number of elements (the *size*) of a finite set E;
- $\|\cdot\|, \langle\cdot;\cdot\rangle$, standard norm and inner product in $\mathbb{R}^2$;
- $S(E)$, area of the measurable set $E \subset \mathbb{R}^2$;
- s, number of the angularly equal sectors S_i^k in M_i;
- $\oplus, \ominus$, addition and subtraction modulo s;
- $\vec{e}(\alpha)$, unit vector with the polar angle α;
- $\mathfrak{C}_i := \{\boldsymbol{p} : \|\boldsymbol{p} - \boldsymbol{r}_i\| = R_{\text{sen}}\}$, circle bounding M_i;
- $\vec{s}_k := \vec{e}[2\pi k/s]$, unit vector along the ray separating the sectors S_i^k and $S_i^{k\ominus 1}$;
- $\vec{m}_k = R_{\text{sen}}\,\vec{e}\left[\frac{2\pi k+\pi}{s}\right]$, vector bisecting S_i^k;
- $\boldsymbol{t}_i^k := \boldsymbol{r}_i + \vec{m}_k$, so-called *top* of the sector S_i^k;
- $\mathbf{sec}_i(\boldsymbol{p})$, index k of the sector S_i^k containing $\boldsymbol{p} \in M_i$.

If point $\boldsymbol{p}$ lies on the border of two sectors, $\mathbf{sec}_i(\boldsymbol{p})$ is the index of the counterclockwise one.

3 Proposed algorithms

They assume that the robots have access to a common direction, which can be acquired, e.g., by using a compass. Any robot i uses this direction as that of the x-axis to build a right-handed Cartesian frame $\mathcal{F}_i$ centered at $\boldsymbol{r}_i$. Also, robot i partitions its monitoring zone into s angularly equal sectors $S_i^0, \ldots, S_i^{s-1}$ rooted at $\boldsymbol{r}_i$ and labeled counterclockwise; see Fig. 1(a), where $s = 6$. The design parameter $s = 3, 4, \ldots$ is common for all robots, and the sector S_i^0 is clockwise edged by the x-axis.

Our algorithms use the following two parameters:

$C > 0$, gain of the control loop;
$s \geq 3$, number of the sectors.

3.1 Sectoring algorithm (*SA*)

At any time $t \geq 0$ any robot i does the following:

s.1) Finds the locations of visible robots (*neighbors*):

$$\mathcal{N}_i := \{\boldsymbol{r}_j : 0 < \|\boldsymbol{r}_j - \boldsymbol{r}_i\| \leq R_{\text{sen}}\}; \tag{2}$$

s.2) Partitions $\mathcal{N}_i$ into *sectorial bins* $B_i^0, \ldots, B_i^{s-1}$:

$$B_i^k = \{\boldsymbol{r}_j : \boldsymbol{r}_j \in \mathcal{N}_i \text{ and } \mathbf{sec}_i(\boldsymbol{r}_j) = k\}; \tag{3}$$

s.3) If a bin B_i^k is empty, concocts a fictitious robot at the "top" $\boldsymbol{t}_i^k$ of the sector S_i^k and "puts" $\boldsymbol{t}_i^k$ into B_i^k;

s.4) Selects the nearest neighbor in every sectorial bin:

$$\boldsymbol{p}_i^k = \arg\min_{\boldsymbol{p} \in B_i^k} \|\boldsymbol{p} - \boldsymbol{r}_i\| \tag{4}$$

and determines its relative position: $\boldsymbol{n}_i^k := \boldsymbol{p}_i^k - \boldsymbol{r}_i$;

s.5) Finds the mean distance to the nearest neighbors:

$$\mathcal{D}_i := s^{-1} \sum_{k=0}^{s-1} \|\boldsymbol{n}_i^k\|; \tag{5}$$

s.6) Computes the current control:

$$u_i := C \cdot \sum_{k=0}^{s-1} \frac{\boldsymbol{n}_i^k}{\|\boldsymbol{n}_i^k\|} \cdot \frac{\|\boldsymbol{n}_i^k\| - \mathcal{D}_i}{\mathcal{D}_i}. \tag{6}$$

If in s.4), a robot faces many nearest neighbors, it tries to make a choice so that the result depends on time continuously, and chooses at random in the case of failure.

In (6), the unit vector $\frac{\boldsymbol{n}_i^k}{\|\boldsymbol{n}_i^k\|}$ is aimed at the nearest neighbor in the sector S_i^k. The entire k-th addend tries to drag the robot either toward the nearest neighbor or away from it depending on whether the distance to this neighbor is above or below the mean of such distances over all beans. The "drag strength" is proportional to the relative deviation of this distance from the mean value. Finally, averaging of the results over all bins is employed.

3.2 Continuous sectoring algorithm (*CSA*)

SA employs a discontinuous regulation rule. Potentially (but not necessarily), this creates risks of unwanted complications: chattering in possible sliding-mode regimes and excessive activity in the control loop.

CSA modifies **SA** in order to reduce the exposure to losses of continuity. As will be shown, not only this goal is achieved but also some extra benefits result from this modification. However, the price for these is a modestly increased computational complexity. So in practice, it is recommended to replace **SA** with **CSA** only if the above complications are proven to be a real concern.

The following events may (not inevitably) cause abrupt jumps of the control signal when executing **SA** :

✧ neighbors leave the monitoring zone;

✧ neighbors move from one sector to another;

✧ Within the same bin, the status of the nearest neighbor passes from one robot to another so that the bearing of the nearest neighbor instantaneously alters.

The measures taken to inhibit jumps are as follows:

❍ Any robot is complemented by its two *reflections* about the boundary rays of the hosting sector;

❍ The concept of a "shadow" of a point is introduced so that the shadow evolves continuously in some cases where the parent point jumps;

❍ The set (2) of neighbors is replaced by the set of their shadows and the shadows of their reflections;

❍ The true nearest neighbor (4) is replaced by something similar that continuously evolves with time.

To come into details, we start with two procedures that are to be executed by any robot i, preliminarily denoting

$$\mathbf{sec}_i^+(\boldsymbol{p}) := \mathbf{sec}_i(\boldsymbol{p}) \oplus 1, \quad \mathbf{sec}_i^-(\boldsymbol{p}) := \mathbf{sec}_i(\boldsymbol{p}) \ominus 1.$$

Building the *reflections* of a point $\boldsymbol{p} \in \boldsymbol{M_i}, \boldsymbol{p} \neq \boldsymbol{r}_i$. The sector $S_i(\boldsymbol{p}) := S_i^{\mathbf{sec}_i(\boldsymbol{p})}$ hosting $\boldsymbol{p}$ is identified and the following steps are performed (see Fig. 1(b)):

Step 1: The distances $D^\pm$ from point $\boldsymbol{p}$ to the boundary rays of $S_i(\boldsymbol{p})$ are computed:

$$D^+ := \underbrace{\sqrt{\|\boldsymbol{p}-\boldsymbol{r}_i\|^2 - \left\langle \vec{s}_{\mathbf{sec}^+(\boldsymbol{p})}; \boldsymbol{p}-\boldsymbol{r}_i\right\rangle^2}}_{\min_{\varsigma\geq 0}\left\|\boldsymbol{r}_i+\varsigma\vec{s}_{\mathbf{sec}^+(\boldsymbol{p})}-\boldsymbol{p}\right\|},$$

$$D^- := \sqrt{\|\boldsymbol{p}-\boldsymbol{r}_i\|^2 - \left\langle \vec{s}_{\mathbf{sec}(\boldsymbol{p})}; \boldsymbol{p}-\boldsymbol{r}_i\right\rangle^2};$$

Step 2: They are converted into the relative values: $\theta^\pm := \frac{D^\pm}{D^+ + D^-} \geq 0, \theta^- + \theta^+ = 1$;

Step 3: The radius-vectors of the *semi-reflections* of $\boldsymbol{p}$ are build via the convex combination of a specially chosen bisector and "boundary" vectors:

$$\widetilde{R}_i^\pm(\boldsymbol{p}) = \theta^\pm \cdot \vec{m}_{\mathbf{sec}^\pm(\boldsymbol{p})} + \theta^\mp \cdot \vec{\zeta}^\pm, \quad \text{where}$$

$$\vec{\zeta}^+ := \|\boldsymbol{p}-\boldsymbol{r}_i\| \cdot \vec{s}_{\mathbf{sec}^+(\boldsymbol{p})}, \quad \vec{\zeta}^- := \|\boldsymbol{p}-\boldsymbol{r}_i\| \cdot \vec{s}_{\mathbf{sec}(\boldsymbol{p})};$$

Step 4: Two *reflections* of $\boldsymbol{p}$ are found:

$$R_i^\pm(\boldsymbol{p}) = \boldsymbol{r}_i + \frac{\widetilde{R}_i^\pm}{\|\widetilde{R}_i^\pm\|} \cdot \max\left\{\|\boldsymbol{p}-\boldsymbol{r}_i\|, \|\tilde{R}_i^\pm\|\right\}. \quad (7)$$

It is easy to see by inspection that the following holds.

Remark 1. *As point $\boldsymbol{p} \neq \boldsymbol{r}_i$ ranges over S_i^k, its reflection $R_i^+(\boldsymbol{p})$ runs in the adjacent half-sector of $S_i^{k\oplus 1}$. Points $\boldsymbol{p}$ and $R_i^+(\boldsymbol{p})$ fuse as $\boldsymbol{p}$ goes to the counterclockwise boundary ray of S_i^k; see Fig. 2(a). As $\boldsymbol{p}$ approaches the other boundary ray, the reflection $R_i^+(\boldsymbol{p})$ goes to the top $\boldsymbol{t}_i^k$ of $S_i^{k\oplus 1}$. Finally, $R_i^+(\boldsymbol{p})$ is a Lipschitz continuous function of $\boldsymbol{p} \neq \boldsymbol{r}_i$. Similar properties hold for $R_i^-(\boldsymbol{p})$.*

Building the *shadow* of a point $\boldsymbol{p} \in \boldsymbol{M_i}, \boldsymbol{p} \neq \boldsymbol{r}_i$:

Step 1: The unit vector $\vec{n}$ is built that is normal to the bisector of $S_i^{\mathbf{sec}_i(\boldsymbol{p})}$ and is oriented counterclockwise;

Step 2: For $\boldsymbol{p}$, the signed distance D_{bis} to this bisector and the distance D_{out} to the border of M_i are calculated:

$$D_{\text{bis}} := \langle \vec{n}_i(\boldsymbol{p}); \boldsymbol{p}-\boldsymbol{r}_i\rangle, \quad D_{\text{out}} := R_{\text{sen}} - \|\boldsymbol{p}-\boldsymbol{r}_i\|;$$

Step 3: The mismatch $\mu := |D_{\text{bis}}| - D_{\text{out}}$ is evaluated;

Step 4: The vector of direction to the shadow is found:

$$\widetilde{\mathcal{S}}_i(\boldsymbol{p}) := \boldsymbol{p} - \boldsymbol{r}_i - \begin{cases} \mu \cdot \vec{n}_i(\boldsymbol{p})\mathbf{sgn}\, D_{\text{bis}} & \text{if } |D_{\text{bis}}| > D_{\text{out}}, \\ 0 & \text{otherwise;} \end{cases}$$

Step 5: The *shadow* of $\boldsymbol{p}$ is build via going from $\boldsymbol{r}_i$ in the just found direction to a position equidistant with $\boldsymbol{p}$:

$$\mathcal{S}_i(\boldsymbol{p}) = \boldsymbol{r}_i + \frac{\widetilde{\mathcal{S}}_i(\boldsymbol{p})}{\|\widetilde{\mathcal{S}}_i(\boldsymbol{p})\|}\|\boldsymbol{p}-\boldsymbol{r}_i\|. \quad (8)$$

It is easy to see that the following is true.

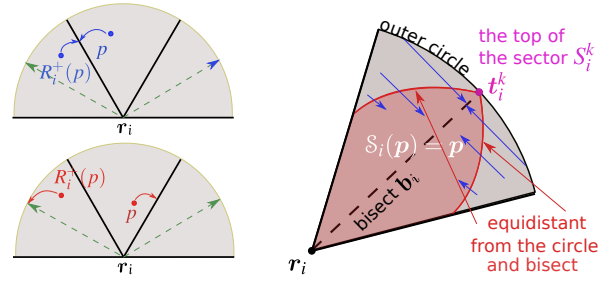


Figure 2. (a) Behavior of the reflections; (b) Building the shadows.

Remark 2. *The shadow $\mathcal{S}_i(\boldsymbol{p})$ is a Lipschitz continuous function of $\boldsymbol{p} \in S_i^k \setminus \boldsymbol{r}_i$ and lies no farther from the bisect $\boldsymbol{b}_i$ of S_i^k than $\boldsymbol{p}$ does. If $\boldsymbol{p}$ is closer to the bisect than to the border $\mathfrak{C}_i$ of M_i (such $\boldsymbol{p}$'s form the light red set in Fig. 2(b)), then $\mathcal{S}_i(\boldsymbol{p}) = \boldsymbol{p}$. Otherwise, $\boldsymbol{p}$ first moves closer to $\boldsymbol{b}_i$ along the normal $\vec{n}$ until its distance to $\boldsymbol{b}_i$ becomes equal to the original distance from $\boldsymbol{p}$ to $\mathfrak{C}_i$; in doing so, $\boldsymbol{p}$ goes away from $\mathfrak{C}_i$. Then the radius vector of $\boldsymbol{p}$ is scaled to restore the initial distance to $\mathfrak{C}_i$. If originally $\boldsymbol{p} \in \mathfrak{C}_i$, then $\mathcal{S}_i(\boldsymbol{p})$ is the top of the sector.*

Appointing to the role of the *nearest neighbor*. Given a finite nonempty set $P \subset S_i^k$ of contenders, the role is entrusted to a point, which continuously depends on the elements of P but may lie outside P. If $P \subset \mathfrak{C}_i$, the role is given to the top $\boldsymbol{t}_i^k$ (see Fig. 2(b)). Otherwise,

nn.1) The weight of every point $\boldsymbol{p} \in P$ is first found:

$$w_i(\boldsymbol{p}|P) := \frac{(R_{\text{sen}} - \|\boldsymbol{p} - \boldsymbol{r}_i\|)^2}{\sum_{\boldsymbol{p}' \in P}(R_{\text{sen}} - \|\boldsymbol{p}' - \boldsymbol{r}_i\|)^2}; \quad (9)$$

nn.2) The relative location of the appointed "nearest neighbor" is defined as the convex combination

$$\boldsymbol{n}_i(P) := \sum_{\boldsymbol{p} \in P} w_i(\boldsymbol{p}|P)(\boldsymbol{p} - \boldsymbol{r}_i). \quad (10)$$

The closer $\boldsymbol{p}$ to robot i, the larger its weight (9). Thus contribution of the close neighbors is prioritized, while neglecting neither member of P.

Description of CSA . This algorithm uses the parameters C and s of **SA** .

At any time $t \geq 0$ any robot i does the following:

c.1) Replaces every element of the set (2) by its shadow and the shadows of its reflections, thus forming

$$\mathfrak{N}_i := \Big\{ \mathcal{S}_i(\boldsymbol{r}_j),\, \mathcal{S}_i\big[R_i^{\pm}(\boldsymbol{r}_j)\big] : j \in \mathcal{N}_i \Big\}; \quad (11)$$

c.2) Like in (3), distributes $\mathfrak{N}_i$ into s sectorial bins $\mathfrak{B}_i^k$;
c.3) Puts the top of the sector into every empty bin;
c.4) In every bin $\mathfrak{B}_i^k$, finds the radius vector $\boldsymbol{n}_i^k := \boldsymbol{n}_i(\mathfrak{B}_i^k)$ of the "appointed" nearest neighbor (10);
c.5) Completes computation of the control u_i just as **SA** does, i.e., in accordance with (5) and (6).

The following proposition shows that **CSA** does get rid of the discontinuities that are characteristic for **SA** .

Proposition 3.1. *Suppose that $s \geq 3$. For any robot i, the control input u_i generated by **CSA** is a Lipschitz continuous function of the team's state $\mathfrak{R} := \{\boldsymbol{r}_i\}_{i=1}^N$.*

Proof. Due to (5) and (6), it suffices to show that for any k, the output $\boldsymbol{n}_i^k$ of the step **c.4) a)** is nonzero and **b)** is a Lipschitz continuous functions of $\mathfrak{R}$.

a) Since $\boldsymbol{r}_j \neq \boldsymbol{r}_i \ \forall j \in \mathcal{N}_i$ by (2), all elements of the set (11) differ from $\boldsymbol{r}_i$ by (7) and (8). Due to **c.2)** and **c.3)**, any sectorial bin is nonempty, contains only nonzero vectors, and lies in a sector whose angular span does not exceed 120° (so long as $s \geq 3$). So the convex combination (10) (with $P := \mathfrak{B}_i^k$) is nonzero.

b) The rules **nn.1), nn.2)** and Rem. 1, 2 imply that $\boldsymbol{n}_i^k$ is a Lipschitz continuous function of $\mathfrak{R}$ near any state at which none of the following circumstances holds:

1. Some robot $\boldsymbol{r}_j$ lies on the boundary circle $\mathfrak{C}_i$;
2. Some robot $\boldsymbol{r}_j$ lies on a boundary radius of a sector;
3. An element of the set (11) lies on such a radius.

It remains to study $\boldsymbol{n}_i^k(\cdot)$ near states $\mathfrak{R}_*$ at which 1), or 2), or 3) holds. We'll examine these cases separately.

1) As $\mathfrak{R}$ wanders near $\mathfrak{R}_*$, the set (2) may change via either accepting j or withdrawing it. When this happens, $\boldsymbol{r}_j \in \mathfrak{C}_i$. Then $\mathfrak{C}_i$ contains any reflection $\boldsymbol{r}_{\boldsymbol{f}}$ of $\boldsymbol{r}_j$ and by Rem. 1, the shadows of $\boldsymbol{r}_{\boldsymbol{f}}$ and $\boldsymbol{r}_j$ are the tops of the respective sectors. In (10) (with $P := \mathfrak{B}_i^k$), only these shadows may jump. By (9), their weights are zero for $\mathfrak{R} = \mathfrak{R}_*$ and smoothly depend on $\mathfrak{R}$ if $\boldsymbol{r}_j$ goes to or from $\mathfrak{C}_i$. Hence Lipschitz continuity is not violated.

2) Building the shadow of $\boldsymbol{p}$ uses the bisect of the sector $S_i^{\mathbf{sec}_i(\boldsymbol{p})}$ containing $\boldsymbol{p}$. If $\boldsymbol{p}$ lies on the common radius of two sectors S_i^k and $S_i^{k\oplus1}$, then $\mathbf{sec}_i(\boldsymbol{p}) = k \oplus 1$. All steps of shadow-building are well defined if using the bisect of the other sector S_i^k. We denote their result and the true shadow by $\mathcal{S}_i^k(\boldsymbol{p})$ and $\mathcal{S}_i^{k\oplus1}(\boldsymbol{p})$, respectively. Then $\mathcal{S}_i^k(\boldsymbol{p})$ is a Lipschitz continuous function of $\boldsymbol{p} \in \overline{S_i^k} \setminus \boldsymbol{r}_i$.

Let $\mathfrak{R}$ wanders near $\mathfrak{R}_* = \{\boldsymbol{r}_j^*\}$. Robot j may move between two adjacent sectors (say S_i^k and $S_i^{k\oplus1}$), going through a point $\boldsymbol{r}_j^*$ of their common radius. Due to 1) of the proof, it suffices to focus on the case where $\boldsymbol{r}_j^* \notin \mathfrak{C}_i$.

Let $\boldsymbol{r}_j \in S_i^k$ goes to $\boldsymbol{r}_j^*$. Then it is easy to see that

$$\begin{aligned}
&R_i^+(\boldsymbol{r}_j) \to \boldsymbol{r}_j^* \text{ and } R_i^+(\boldsymbol{r}_j) \in S_i^{k\oplus1} \text{ by Rem. 1;}\\
&R_i^-(\boldsymbol{r}_j) \to \boldsymbol{t}_i^{k\ominus1} \text{ and } R_i^-(\boldsymbol{r}_j) \in S_i^{k\ominus1} \text{ by Rem. 1;}\\
&\mathcal{S}_i(\boldsymbol{r}_j) = \mathcal{S}_i^k(\boldsymbol{r}_j) \to \mathcal{S}_i^k(\boldsymbol{r}_j^*);\\
&\mathcal{S}_i\big[R_i^+(\boldsymbol{r}_j)\big] = \mathcal{S}_i^{k\oplus1}\big[R_i^+(\boldsymbol{r}_j)\big] \to \mathcal{S}_i^{k\oplus1}(\boldsymbol{r}_j^*);\\
&\mathcal{S}_i\big[R_i^-(\boldsymbol{r}_j)\big] = \mathcal{S}_i^{k\ominus1}\big[R_i^-(\boldsymbol{r}_j)\big] \to \boldsymbol{t}_i^{k\ominus1};
\end{aligned}$$

Meanwhile, as $\boldsymbol{r}_j$ goes to $\boldsymbol{r}_j^*$ while remaining in $S_i^{k\oplus1}$,

$$\begin{aligned}
&R_i^-(\boldsymbol{r}_j) \to \boldsymbol{r}_j^* \text{ and } R_i^-(\boldsymbol{r}_j) \in S_i^k \text{ by Rem. 1;}\\
&R_i^+(\boldsymbol{r}_j) \to \boldsymbol{t}_i^{k\oplus2} \text{ and } R_i^+(\boldsymbol{r}_j) \in S_i^{k\oplus2} \text{ by Rem. 1;}\\
&\mathcal{S}_i(\boldsymbol{r}_j) = \mathcal{S}_i^{k\oplus1}(\boldsymbol{r}_j) \to \mathcal{S}_i^{k\oplus1}(\boldsymbol{r}_j^*);\\
&\mathcal{S}_i\big[R_i^+(\boldsymbol{r}_j)\big] = \mathcal{S}_i^{k\oplus2}\big[R_i^+(\boldsymbol{r}_j)\big] \to \boldsymbol{t}_i^{k\oplus2};\\
&\mathcal{S}_i\big[R_i^-(\boldsymbol{r}_j)\big] = \mathcal{S}_i^{k}\big[R_i^-(\boldsymbol{r}_j)\big] \to \mathcal{S}_i^{k}(\boldsymbol{r}_j^*).
\end{aligned}$$

Thus as $\boldsymbol{r}_j \to \boldsymbol{r}_j^*$, the set (11) converges to one of two limit sets, depending on the "side" from which $\boldsymbol{r}_j$ goes to $\boldsymbol{r}_j^*$. These limit sets have common points $\mathcal{S}_i^k(\boldsymbol{r}_j^*)$ and $\mathcal{S}_i^{k\oplus1}(\boldsymbol{r}_j^*)$ and may differ only by elements of the form $\boldsymbol{t}_i^{k\ominus1}, \boldsymbol{t}_i^{k\oplus2}$. Since in (10), the latter elements are taken with the zero weights (9) and the weights are Lipschitz continuous functions of the point, we infer that $\boldsymbol{n}_i^k$ is such a function of the state $\mathfrak{R}$.

3) It suffices to note that this situation cannot occur if both 1) and 2) do not hold.

4 Performance metrics

There are practical reasons to evaluate the quality of area coverage from a variety of aspects. So it does not come as a surprise that many performance metrics have been proposed in the literature. We employ not a single but a whole series of them, thus attempting to carry out multifaceted analysis of the concerned algorithms.

Elapsed time [Heo and Varshney, 2003] The time elapsed until all robots halt.

Coverage percentage [Heo and Varshney, 2003] is the portion of the area covered by the robots altogether:

$$\textbf{Coverage\_percentage} := \frac{S(\bigcup_{i=1}^{N} M_i)}{S(\mathcal{R})}. \qquad (12)$$

Mean distance [Heo and Varshney, 2003] results from averaging the distances traveled by the robots.

Distances uniformity [Heo and Varshney, 2003] this index results from averaging (over all robots) the dispersion of the distances to the robot's neighbors:

$$\textbf{Distances\_uniformity} := \\ := \frac{1}{N}\sum_{i=1}^{N}\sqrt{\frac{1}{|\mathcal{N}_i|}\sum_{j\in\mathcal{N}_i}(d_{ij}-M_i)^2}, \\ \text{where } d_{ij} := \|\boldsymbol{r}_i - \boldsymbol{r}_j\|, \quad M_i := \frac{1}{|\mathcal{N}_i|}\sum_{j\in\mathcal{N}_i} d_{ij}.$$

Energy consumption [Ghahroudi et al., 2019] is the total energy spent by all robots provided that one unit of energy is spent to move a robot by one unit of length.

We also introduce a new criterion to assess the amount of variation in the number of robots that detect a particular point, where the variation is registered as the point runs all over the region $\mathcal{R}$ to be covered.

Area coverage uniformity. We denote by $C(\boldsymbol{r}) = |\{j\colon \boldsymbol{r} \in M_j\}|$ the number of robots that detect $\boldsymbol{r}$ ($C(\boldsymbol{r}) := 0$ if there are no such robots), and put

$$\textbf{Area\_uniform} := \sqrt{\frac{1}{S(\mathcal{R})}\int_{\mathcal{R}}[C(\boldsymbol{r}) - C_{\text{mean}}]^2\, d\boldsymbol{r}},$$

Where $C_{\text{mean}} := \frac{1}{S(\mathcal{R})}\int_{\mathcal{R}} C(\boldsymbol{r})\, d\boldsymbol{r}$. The index C_{mean} evaluates the mean number of robots that must be disabled to reduce coverage percentage. A high **Area_uniform** value indicates the presence of weak spots in the network - areas with either low or high robot concentrations. Low-concentration spots are more vulnerable: disabling fewer robots in these areas can compromise coverage of $\mathcal{R}$. High-concentration spots are undesirable as they present attractive targets for attackers capable of disabling unlimited robots within a specific radius.

Thus, **Area_uniform** focuses on vulnerability issues. Uniformity of the team's distribution between given subregions can be assessed by the index called the deployment entropy. Unlike the former, the latter depends on partition of the scene into cells, which is typically not a part of the scenario but is an artifice of the researcher.

5 Analysis of the performance of the algorithms

To highlight motivation and demonstrate benefits of the control protocols proposed in Sec. 3.1 and 3.2, we compare their performance with that of several well established algorithms applicable in the scenario considered in this paper. For the convenience of the reader, we start by outlining these algorithms; for more details about them, we refer the reader to the original works.

5.1 VFB algorithms of area coverage.

Distributed Self Spreading Algorithm (DSSA) [Heo and Varshney, 2003] handles the discrete-time model:

$$\boldsymbol{r}_i(t+1) = \boldsymbol{r}_i(t) + u_i(t), \quad t = 0, 1, \ldots. \qquad (13)$$

Its parameters: O_{lim}, S_{lim} – bounds on the numbers of the so-called oscillatory and "marking time" steps; e – threshold used to judge the type of the step.

Any robot i at $t = 0$ starts counting the numbers of oscillatory and "marking time" steps: $O_i := 0$, $S_i := 0$.

At any time t this robot does the following:

1. Sets the expected nodes' density: $\mu = \frac{N\cdot\pi\cdot R_{\text{sen}}^2}{S(\mathcal{R})}$;
2. Finds the number $D_i(t) = |\mathcal{N}_i(t)|$ of the neighbors, where the set of them includes i and is defined by
$$\mathcal{N}_i(t) := \{j\colon \|\boldsymbol{r}_j(t) - \boldsymbol{r}_i(t)\| < R_{\text{sen}}\}; \qquad (14)$$
3. Finds a virtual force from every neighbor $j \in \mathcal{N}_i(t)$:
$$F_{ij}(t) := \frac{D_i(t)}{\mu^2}(\|\boldsymbol{r}_{ij}(t)\| - R_{\text{sen}})\frac{\boldsymbol{r}_{ij}(t)}{\|\boldsymbol{r}_{ij}(t)\|}, \qquad (15)$$
$$\text{where } \boldsymbol{r}_{ij}(t) := \boldsymbol{r}_j(t) - \boldsymbol{r}_i(t);$$
4. Finds the "precontrol": $\tilde{u}_i(t) = \sum_{j\in\mathcal{N}_i(t)} F_{ij}(t)$;
5. Checks the prospective step for its oscillatory status:
if $|\boldsymbol{r}_i(t-1) - \boldsymbol{r}_i(t) - \tilde{u}_i(t)| < e$, then $O_i := O_i + 1$;
if $O_i \geq O_{\text{lim}}$, then $u_i(t) := \frac{\tilde{u}_i(t)}{2}$ and goes on to 8);
6. Checks this step for its "marking time" status:
if $|\tilde{u}_i(t)| < e$, then $S_i := S_i + 1$
if $S_i \geq S_{\text{lim}}$, then $u_i(t) := 0$ and moves on to 8);
7. $u_i(t) := \tilde{u}_i(t)$, upd. $\boldsymbol{r}_i(t+1)$, $t := t+1$, goes to 2);
8. Updates $\boldsymbol{r}_i(t+1)$, $t := t+1$, and halts.

Further, the argument t is omitted for brevity.

Self-organizing node deployment algorithm (SODA) [Ghahroudi et al., 2018] also handles the model (13). **SODA** is identical to **DSSA**, except for one point: in (15), $\frac{\min\{D_i,\mu\}}{\mu^2}$ is put in place of $\frac{D_i}{\mu^2}$.

Smart self-organizing node deployment algorithm (SSND) [Ghahroudi et al., 2019] comes to execution of **DSSA** for only a time-varying subset of nodes that is formed via computing a special variable Elig_i (called *eligibility*), with using a free parameter $\alpha > 0$.

Any robot i at $t = 0$ initializes counting $m_i := 0$ the number of times when i was elected to move.

At any time t this robot does the following:

1. Executes **DSSA** to calculate $\tilde{u}_i$;
2. Finds $|\mathcal{N}_i^{lf}|$, where $\mathcal{N}_i^{lf} := \{j\colon j \in \mathcal{N}_i, \tilde{u}_i \geq \tilde{u}_j\}$;
3. Calculates the "degree of eligibility": $\text{Elig}_i := \frac{\alpha\cdot(|\mathcal{N}_i^{lf}| - m_i) - |\mu - D_i|}{N}$, where μ and D_i are defined in 1) and 2) of **DSSA**;
4. If $\text{Elig}_i \geq \max_{j\in\mathcal{N}_i}\text{Elig}_j$, then $m_i := m_i + 1$ and robot i moves according to the instructions of **DSSA**; otherwise, the robot stands still.

SWARM [Mathews et al., 2012] (biologically inspired swarm robotic network coverage algorithm uses the model (13) and parameters w_1, w_2, w_3 such that

$$w_1 + w_2 + w_3 = 1, \qquad w_1 \geq 0,\ w_2 \geq 0,\ w_3 \geq 0.$$

At any time $t \geq 0$ any robot i does the following:

1. Finds the size $|\mathcal{N}_i|$ of the set (14);
2. Forms a finite set $\mathcal{O}_i$ of points uniformly distributed along the boundaries of the visible obstacles;
3. Calculates $F_i^{\text{sep}} := \frac{1}{|\mathcal{N}_i|} \cdot \sum_{j \in \mathcal{N}_i} \frac{\sqrt{3}(\boldsymbol{r}_j - \boldsymbol{r}_i)}{2|\boldsymbol{r}_j - \boldsymbol{r}_i|^3}$, $F_i^{\text{coh}} := \left(\sum_{j \in \mathcal{N}_i} \boldsymbol{r}_j\right) - \boldsymbol{r}_i$, $F_i^{\text{alig}} := \frac{1}{|\mathcal{N}_i|} \cdot \sum_{j \in \mathcal{N}_i} u_j(t-1)$, $F_i^{\text{obst}} := \frac{1}{|\mathcal{O}_i|} \cdot \sum_{\boldsymbol{p} \in \mathcal{O}_i} \frac{\sqrt{3}(\boldsymbol{p} - \boldsymbol{r}_i)}{2|\boldsymbol{p} - \boldsymbol{r}_i|^2}$;
4. Computes a "virtual force" $u_i = w_1 \cdot (F_i^{\text{sep}} + F_i^{\text{obs}}) + w_2 \cdot F_i^{\text{coh}} + w_3 \cdot F_i^{\text{alig}}$ in (13);
5. Updates $\boldsymbol{r}_i(t+1)$, $t := t + 1$, and returns to 1).

Virtual Spring Force Algorithm (VFA-SF) [Deng et al., 2019] handles a continuous-time model and mobile robots with the double integrator kinematics

$$\frac{d^2 \boldsymbol{r}_i(t)}{dt^2} = u_i(t) \qquad \forall t \geq 0.$$

The method uses the following notations and parameters:
- $\varkappa > 0$, $D_m > 0$, parameters of a virtual spring;
- $d_{ij}(t) := \|\boldsymbol{r}_j - \boldsymbol{r}_i\|$, $\boldsymbol{r}_{ij} := \varkappa(\boldsymbol{r}_j - \boldsymbol{r}_i)/d_{ij}$;
- $\gamma \geq 0$, damping (friction) parameter;
- F_{centri}, centripetal force parameter.

At any time t any robot i does the following:

1. Finds the (unordered) set of neighbors (14);
2. Forms the set $\Phi_i(t)$ of all neighbors $j \in \mathcal{N}_i(t)$ s.t. either $\|\boldsymbol{r}_j - \boldsymbol{r}_i\| \leq \|\boldsymbol{r}_{j'} - \boldsymbol{r}_i\|\ \forall j' \in \mathcal{N}_i(t)$ or angle between $\boldsymbol{r}_{j'} - \boldsymbol{r}_i$ and $\boldsymbol{r}_j - \boldsymbol{r}_i$ exceeds $60°$;
3. Defines the control input as a virtual force:

$$u_i(t) := F_i^{\text{e}}(t) + F_i^{\text{f}}(t) + F_i^{\text{ce}}(t), \quad \text{where}$$

$$F_i^{\text{e}}(t) = \sum_{j \in \Phi_i} (d_{ij} - D_m)\boldsymbol{r}_{ij},\ F_i^{\text{ce}}(t) = -F_{\text{centri}} \cdot \boldsymbol{r}_i,$$

$$F_i^{\text{f}}(t) = \begin{cases} -F_i^{\text{e}} - F_i^{\text{ce}} & \text{if } \frac{d\boldsymbol{r}_i}{dt} \approx 0, F_i^{\text{e}} + F_i^{\text{ce}} \approx 0, \\ -\gamma \frac{d\boldsymbol{r}_i}{dt} & \text{otherwise.} \end{cases}$$

5.2 Experimental setup

In our tests, robots are initially distributed randomly within a 1 m radius disc. For each of the four examined scenes and seven algorithms, we perform nine test series. The number of robots varies from 10 to 50 in steps of 5 across series. Each series consists of 100 tests with different initial robot deployments. Performance index values are averaged across all tests within a series, with results shown in Fig. 4, 5, and 6. Series are identified by the number of robots plotted on the abscissa axis, yielding 6300 total tests per scene-algorithm combination.

Scene boundaries and obstacles are modeled as finite chains of evenly distributed "fictitious" stationary robots. The motion of actual robots is affected by these fictitious counterparts since actual robots perceive and process fictitious robots identically to other robots.

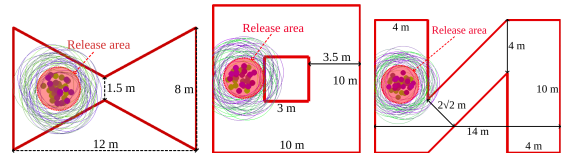
Robot speed is upper bounded at $\overline{v} = 4.0$ m/s. For continuous-time algorithms, speed is saturated at this value when exceeded. For discrete-time algorithms, the time step corresponds to $\tau = 20.0$ ms of real time, saturating u_i from (13) at $\overline{v} \cdot \tau = 0.08$ m. Vector $\vec{v}$ saturation at level $L > 0$ is defined as $\vec{v}$ when $\|\vec{v}\| \leq L$, and $L\vec{v}/\|\vec{v}\|$ otherwise.

In all tests, $R_{\text{sen}} = 2.0$ m. The parameters of the algorithms are as follows (except for **SA** and **CSA**, they are taken from the original works):

SA and **CSA**	**DSSA**, **SODA**, **SSND**
$s = 6$, $C = 10$ (for **CSA**), $C = 6$ (for **SA**)	$O_{\text{lim}} = S_{\text{lim}} = 10$, $e = 0.0052$ m, $\alpha = 1$ (for **SSND**)
SWARM	**VFA-SF**
$w_1 = w_2 = 0.4$, $w_3 = 0.2$	$\varkappa = 15$, $D_m = R_{\text{sen}}/\sqrt{3}$, $\gamma = 7.75$, $F_{\text{centri}} = 0.005$

The algorithms **SODA, DSSA**, and **SSND** are close to each other. So in some tests, the difference in their performance is hardly visible. This motivated us not to draw the graphs related to all of these methods in some of the subsequent Fig. 4, 5, 6, 7. In such cases, the retained graph from this group should be viewed as the substitute for the absent ones. In these figures, the distances are measured in meters, time in seconds, and energy in conditional units (as is commented in Sec. 4).

Multimedia of typical tests and complementary material are available at https://drive.google.com/drive/folders/1x_9HW-35Qo9lj3xcoB6eFlb7fy8U9Tso?usp=sharing



(a) Butterfly-like scene (b) Circular corridor (c) Zigzag-shaped corridor

Figure 3. Simulation scenes

5.3 Simulation tests: a butterfly-like scene

It is illustrated in Fig. 3(a) and models the situation of a narrow passage between two chambers. In Fig. 3(a), like in the subsequent Fig. 3(b) and 3(c), the initial locations of the robots are shown as thick dots, the disc of their initial distribution is depicted in light red, and a lot of tiny circles gives the picture of the boundaries of the monitoring zones of all robots at their initial deployment. In Fig. 3(a), the robotic team starts in the left chamber and has to spread over the union of the chambers, thus having to partly penetrate through the narrow passage. The scene in Fig. 3(a) is used in an attempt to assess the capacity of the algorithms to "past bottlenecks" in the environment.

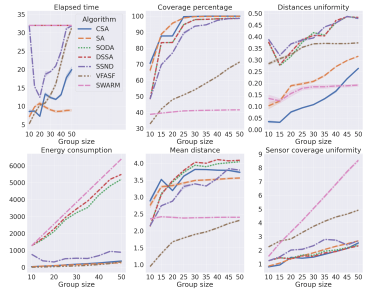


Figure 5. Performance in the circular corridor

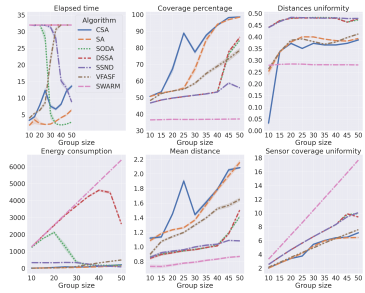


Figure 4. Performance in the butterfly-like scene

By Fig. 4, **CSA** and **SA** dominate for large teams ($\geq$30), with **CSA** superior at $N \in [20, 28]$ and demonstrate near-perfect 100% coverage at max size vs. $\approx$85% for others

Other methods: **VFA-SF** is competitive for $N \leq 45$ (matches **SA** at $N \leq 25$), but overtaken by **SODA/DSSA** for $N > 45$; **SODA/DSSA/SSND** have similar performance for $N \leq 42$ (shared formula (15)), but **SSND** declines for large N; **SWARM** is non-convergent (outsider)

Coverage uniformity (lower=better):

Leaders: **SA**, **CSA**, **VFA-SF**
Intermediate: **SODA**, **DSSA**, **SSND** (gap $\uparrow$ with size)
Outsider: **SWARM**

Convergence (all tests stop at 32 sec):

Convergent: **CSA**, **SA** (**SA** fastest for $N \leq 35$)
Non-convergent: **SODA**, **SSND** (small teams); **VFA-SF** (large teams); **SWARM** (all sizes)

SODA/SSND stop earlier for large teams (explained by poorer coverage).

Distance analysis:

Max travel distance: **CSA/SA** (due to uniform coverage strategy)
Distance uniformity: **CSA**, **SA**, **VFA-SF** (mid-rank)
SWARM leads distance uniformity but has poor coverage

Multimedia of typical tests in the butterfly-like scene are available at

https://drive.google.com/drive/folders/1EaQyaH3X2QzNmDpbWETiw2caB4vbjOeu

5.4 Simulation tests: a circular corridor

In the scene from Fig. 3(b), the robots should move along a circular corridor into two directions, eventually meet somewhere at its middle, and cooperatively cover its entirety. The objective of this experiment is to assess the capacity of the algorithms to seep into all passages, to flow around obstacles, and to unite the robots into an uniform ensemble after various sub-groups of robots meet "on a collision course".

By Fig. 5, **CSA** and **SA** maintain leadership in coverage percentage and uniformity across all team sizes, achieving near-perfect 100% coverage at smaller sizes than in Fig. 4. Unlike previous results, **SODA**, **DSSA**, and **SSND** now provide comparable performance for large teams, while **SWARM** remains weakest and **VFA-SF** shows poorer uniformity.

Convergence analysis reveals:

SWARM fails to converge for any team size
SSND, **DSSA**, **VFA-SF**, and (unlike before) **SODA** struggle with large teams
CSA and **SA** converge for all sizes, dominating for $N \geq 30$
CSA's slower convergence for large N reflects its cautious control strategy

For distance uniformity, algorithms split into two groups: **CSA**, **SA**, and **SWARM** perform best, with **CSA** leading for $N \leq 42$ and **SWARM** for larger teams (unlike Sec. 5.3); others show uniformly worse performance.

As in Sec. 5.3, **SWARM** and **VFA-SF**'s lower coverage corresponds to shorter travel distances.

Multimedia of typical tests in the circular corridor are available at https://drive.google.com/drive/folders/1LjTbvcZ-yPsCJe8F2XocuoZRjxuq2-tL

5.5 Simulation tests: a zigzag-shaped corridor

Fig. 3(c) is concerned with an attempt to assess the algorithms when driving the robots through narrow passages with sharp bends in alternating directions. The tests are ended at 48 sec.

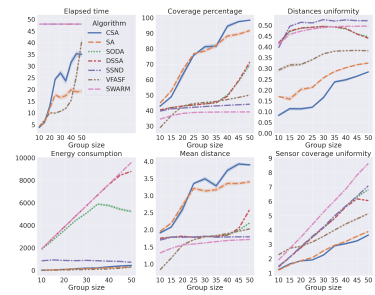


Figure 6. Performance in the zigzag-shaped corridor

According to Fig. 6:

CSA and **SA** dominate coverage percentage and uniformity across all team sizes

Other methods (**SODA, DSSA, SSND, VFA-SF**) show comparable coverage for $N \leq 40$: for $N > 40$: **SODA/DSSA** lead, **SSND** lags; **VFA-SF** leads coverage uniformity for $N \geq 20$
SWARM performs worst in both metrics

Convergence behavior:

SWARM, SODA, DSSA, SSND fail to converge
Convergent methods: **SA, CSA, VFA-SF**
VFA-SF converges fastest for $N \leq 40$ (correlates with poorer coverage)
Surprisingly, **SA/CSA** converge faster than **VFA-SF** for $N > 40$

Distance analysis:

CSA/SA excel in distance uniformity
Their longer travel distances correlate with superior coverage
For $N \geq 25$: **CSA** leads coverage, while **SA** excels in energy/distance efficiency

Multimedia of typical tests in the zigzag shaped corridor are available at

https://drive.google.com/drive/folders/1w9qb8yfQKQfqWiFuWDrho3CUfpP8BkXV

5.6 Simulation tests: averaging over all scenes

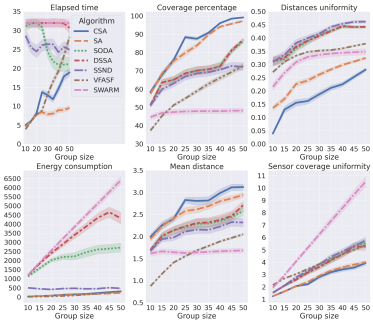


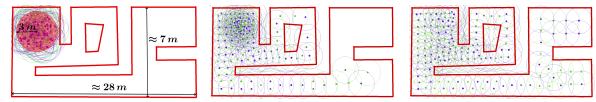
Figure 7. Performance indices averaged over all scenes

The previous analysis, particularly its comparative component, shows that the strengths and weaknesses of the methods depend, to some extent, on scene characteristics. To provide a comprehensive overview, Fig. 7 displays performance index values averaged across all examined scenes.

Regarding coverage percentage, coverage uniformity, and distance uniformity, averaging transforms the established leadership of **SA** and **CSA** in individual scenes into clear dominance over competitors. **CSA** generally outperforms **SA** slightly in coverage percentage and significantly in distance uniformity, while both methods perform similarly in coverage uniformity. However, **CSA** is less efficient (up to twice as slow) in convergence time. Given the previously discussed advantages, this can be viewed as a reasonable trade-off for superior coverage quality.

The mean distance traveled by robots under **SA** and **CSA** exceeds that of other methods, correlating with better coverage quality: these methods drive robots to all critical scene areas, both near and distant, while others may miss certain regions. In energy consumption, **SA** and **CSA** share leadership with **VFA-SF**.

5.7 Testing SA and CSA in a more entangled scene

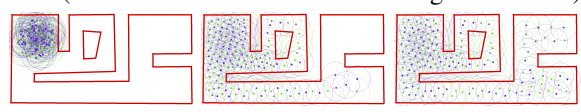


(a) t = 0 s (b) t = 60 s (c) t = 200 s

Figure 8. Behavior of the team driven by **CSA**

The scenes from Sec. 5.3—5.5 are pretty much models of prototypical features of real environments. In this section, we present an evidence that SA and CSA are capable of successfully handling rather entangled scenes, which look like a piece of the layout of a real building. The scene consists of four chambers, one in the form of a circular corridor, and three straight corridors. In this scene, **SA** and **CSA** are tested with 150 robots, eight sectors $s = 8$ in Fig. 1(a), and the gain $C = 6$ in (6).

According to Fig. 8(a), the team is initially disembarked within a disc with a radius of 3 m, which lies in the upper-left chamber. By Fig. 8(b),only the right chambers are not completely covered by the monitoring zones of the robots at $t = 60$ sec. Fig. 8(c) demonstrates that it takes only 200 sec to completely cover the entire scene (with miserable omissions in the right chambers).



(a) t = 0 s (b) t = 24 s (c) t = 96 s

Figure 9. Behavior of the team driven by **SA**

Fig. 9 demonstrates the behavior of the team driven by **SA** and shows that **SA** is twice as fast as **CSA** in achieving similar both intermediate and ultimate results.

6 Conclusion and future work

An evidence is obtained that if coverage or distances uniformity, or coverage percentage are among the key issues, the algorithms **SA** and **CSA** can be recommended for use. They are also worth thinking about in the face of complex environments. On average, **CSA** provides better coverage percentage and distances uniformity than **SA**, though **CSA** is designed with the immediate purpose to take measures against abrupt jumps of the control signal. Meanwhile, **CSA** lags behind **SA** w.r.t. energy consumption and speed, and is more challenging for installation. When choosing between **SA** and **CSA** in practical setting, it is prudent to start with **SA** and, only if the outcome of **SA** is not good enough, to turn to **CSA**.

Our comparative analysis in complex scenes also comprised the algorithms **DSSA** [Heo and Varshney, 2003], **SODA** [Ghahroudi et al., 2018], **SSND** [Ghahroudi et al., 2019], **SWARM** [Mathews et al., 2012], and **VFA-SF** [Deng et al., 2019]. Compared with **CSA** and **SA**, the

methods **DSSA** and **SODA** are less demanding on the sensor equipment since they do not need access of all robots to a common direction by using, e.g., a compass. **VFA-SF** favorably stands out in that it uses the double integrator model of the robots, which is a more authentic model of many real-life systems than the discrete-time models (like those used in **DSSA, SODA, SSND** and **SWARM**) or first-order continuous-time models (like that used in **SA** and **CSA**).

Our future work envisions comparative analysis involving basic Voronoi tessellation-based methods. While this paper assumes all robots access a common fixed reference direction, the proposed algorithms remain directly applicable when each robot has an individual direction. Studying the algorithm's performance under these conditions is underway. Additional ongoing research includes detailed testing of the algorithms on large-scale robotic teams, with sizes inspired by real-world wireless sensor network deployments. Future work will also extend the proposed algorithms to 3D missions and second-order kinematic robots.

7 Acknowledgements

This work was supported by the grant of the state program of the "Sirius" Federal Territory "Scientific and technological development of the "Sirius" Federal Territory" (Agreement N18-03 date 10.09.2024).

References

Bartolini, N., Calamoneri, T., Ciavarella, S., La Porta, T., and Silvestri, S. (2017). Autonomous mobile sensor placement in complex environments. *ACM Trans. on Auton. and Adaptive Syst.*, **12** (2), pp. 1–28.

Cortés, J. and Egerstedt, M. (2017). Coordinated control of multi-robot systems: A survey. *SICE Journal of Control, Measurement, and System Integration*, **10**, pp. 495–503.

Deng, X., Yu, Z., Tang, R., Qian, X., Yuan, K., and Liu, S. (2019). An optimized node deployment solution based on a virtual spring force algorithm for wireless sensor network applications. *Sensors*, **19** (8).

Eledlebi, K., Ruta, D., Hildmann, H., Saffre, F., Al Hammadi, Y., and Isakovic, A. F. (2022). Coverage and energy analysis of mobile sensor nodes in obstructed noisy indoor environment: A Voronoi-approach. *IEEE Transactions on Mobile Computing*, **21** (8), pp. 2745–2760.

Ferri, G., Munafó, A., Tesei, A., Braca, P., Meyer, F., Pelekanakis, K., Petroccia, R., Alves, J., Strode, C., and LePage, K. (2017). Cooperative robotic networks for underwater surveillance: An overview. *IET Radar Sonar and Navigation*, **11** (12), pp. 1740–1761.

Ghahroudi, M., Shahrabi, A., and Boutaleb, T. (2022). A distributed self-organising node deployment algorithm for mobile sensor networks. *International Journal of Communication Systems*, **35** (16), pp. No. 5309.

Ghahroudi, M. S., Shahrabi, A., and Boutaleb, T. (2018). An efficient self-organizing node deployment algorithm for mobile sensor networks. In *UBICOMM 2018, The 12th Int. Conf. on Mobile Ubiquitous Computing, Systems, Services and Technologies*, IARIA.

Ghahroudi, M. S., Shahrabi, A., and Boutaleb, T. (2019). A smart self-organizing node deployment algorithm in wireless sensor networks. In *2019 15th Annual Conference on Wireless On-demand Network Systems and Services (WONS)*, pp. 17–23.

Heo, N. and Varshney, P. (2003). A distributed self spreading algorithm for mobile wireless sensor networks. In *2003 IEEE Wireless Communications and Networking, 2003. WCNC 2003.*, vol. 3, pp. 1597–1602 vol.3.

Houssein, E., Saad, M., Djenouri, Y., Hu, G., Ali, A., and Shabah, H. (2024). Metaheuristic algorithms and their applications in wireless sensor networks: Review, open issues, and challenges. *Cluster Computing*, **27** (10), pp. 13643–13673.

Ma, K., Zhang, Y., and Trappe, W. (2008). Managing the mobility of a mobile sensor network using network dynamics. *IEEE Transactions on Parallel and Distributed Systems*, **19** (1), pp. 106–120.

Mahbub, M. (2020). A smart farming concept based on smart embedded electronics, internet of things and wireless sensor network. *Internet of Things*, **9**.

Mathews, E., Graf, T., and Kulathunga, K. (2012). Biologically inspired swarm robotic network ensuring coverage and connectivity. In *2012 IEEE Int. Conf. on Systems, Man, and Cybernetics*, pp. 84–90.

Muhsen, D., Sadiq, A., and Raheem, F. (2024). A survey on swarm robotics for area coverage problem. *Algorithms*, **17** (3).

Priyadarshi, R., Gupta, B., and Anurag, A. (2022). Deployment techniques in wireless sensor networks: a survey, classification, challenges, and future research issues. *Journal of Supercomputing*, **76** (1).

Sadeghi Ghahroudi, M., Shahrabi, A., Ghoreyshi, S. M., and Alfouzan, F. A. (2023). Distributed node deployment algorithms in mobile wireless sensor networks: Survey and challenges. *ACM Trans. Sen. Netw.*, **19** (4).

Savkin, A., Cheng, T., Xi, Z., Javed, F., Matveev, A., and Nguyen, H. (2015). *Decentralized Coverage Control Problems for Mobile Robotic Sensor and Actuator Networks*. Wiley and IEEE Press, Hoboken, NJ.

Sharma, A. and Lather, J. (2024). Syncronization of coupled quadrocopters using contraction theory. *Cybernetics and physics*, **13** (2), pp. 142–151.

Somov, Y., Butyrin, S., and Somov, S. (2023). Data transfer in a decentralized network of robots using a local voting protocol. *Cybernetics and physics*, **12** (2), pp. 145–151.

Wang, Z., Zhou, S., and Wang, Z. (2018). Underwater distributed antenna systems: Design opportunities and challenges. *IEEE Commun. Mag.*, **56**, pp. 178–185.