Pollutant Spill Containment via Flocking

Luan Rodrigues[1,2], Nadia Nedjah[1,2(✉)], and Luiza de Macedo Mourelle[1,2(✉)]

[1] Department of Electronics Engineering and Telecommunications, State University of Rio de Janeiro, Rio de Janeiro, Brazil
{nadia,ldmm}@eng.uerj.br

[2] Department of Systems Engineering and Computation, State University of Rio de Janeiro, Rio de Janeiro, Brazil

Abstract. The increasing contamination of water bodies by low-density pollutants, such as oil and its derivatives, represents a significant environmental challenge, affecting aquatic ecosystems and coastal economies. Conventional response relies on floating booms that teams must position and maintain, present limitations due to adverse environmental factors like winds and ocean currents. This study proposes an innovative approach based on swarm intelligence that automates the assembly, positioning, and dynamic stabilization of the same type of physical boom, each autonomous surface vehicle is equipped with a short boom segment. Inspired by collective behaviors in nature, the method combines the Flocking algorithm and Particle Swarm Optimization, enabling mobile agents to detect and surround the contaminated area in a distributed and efficient manner. Simulations were conducted to evaluate the effectiveness of the technique in different scenarios, considering pollutant patches with both geometric and abstract shapes. The results demonstrated that the method can form stable barriers, adapting to variations in the shape and initial positions of the agents. Additionally, the approach proved robust against environmental challenges, ensuring effective pollutant containment without the need for direct human intervention.

Keywords: Swarm Intelligence · Pollutant Containment · Multi-Agent Systems · Flocking · PSO · Oil Spill

1 Introduction

Among the various pollutants, oil spills stand out as one of the greatest threats to ecosystems, as fish exposed to oil and its toxic compounds (for example, Polycyclic Aromatic Hydrocarbons, or PAHs) suffer from mutations in eggs, deformities in embryos and larvae, and damage to gills, impairing respiration, mobility, and making these animals unsuitable for consumption [1]. Species with limited mobility, such as crustaceans and benthic mollusks, are even more vulnerable, often accumulating toxins that disrupt their natural growth and reproductive cycles [2].

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2026
O. Gervasi et al. (Eds.): ICCSA 2025 Workshops, LNCS 15888, pp. 37–55, 2026.
https://doi.org/10.1007/978-3-031-97596-7_3

The most commonly used solution for pollutant spills involve the deployment of containment barriers to prevent the substance from spreading in the aquatic environment, as once dispersed, removal becomes nearly unfeasible. However, these measures heavily rely on human intervention and are subject to environmental adversities, such as winds, currents, and tides. These conditions can compromise the efficiency of the barriers, especially when wave heights exceed 1.5 m or when the current speed surpasses 0.5 m/s, leading to failures such as pollutants passing underneath or overflowing the barriers [3]. Additionally, the time required to position these barriers can be significant due to complex logistics and the need for specialized labor, particularly in large-scale spills or hard-to-reach areas. Under unfavorable weather conditions, the pollutant can spread rapidly, exacerbating the problem. Thus, although essential in the initial containment stage, the effectiveness of barriers is closely linked to environmental factors and the speed of their deployment [4].

In this work, the design and simulation of an approach based on swarm intelligence is proposed to enable the automatic containment of these low-density pollutants on the water [5]. Inspired by collective behaviors in nature, such as those observed in fish schools, swarm intelligence offers a scalable, flexible, and resilient solution in which each agent in the swarm collaborates to identify and isolate the contaminant, increasing efficiency compared to traditional methods, while also reducing operational costs and minimizing environmental impacts [6].

The primary objective is to propose a scalable, stable, and computationally low-cost method that efficiently identifies the boundary between areas of high and low pollutant concentration. This method is mainly based on the flocking behavior and should ensure a reduced convergence time. Furthermore, once the boundary is detected, the agents would be distributed to form a uniform and evenly spaced barrier to encapsulate the pollutant. It is intended that this barrier remains stable and in position, regardless of environmental conditions.

This paper is structured into five main sections. Section 2 describes related works addressing the problem of boundary identification and barrier formation using swarm intelligence. Section 3 details the methods required for the development of the control approach. The derivation of the proposed method is presented in Sect. 4. Simulation results and experiments are discussed in Sect. 5. Finally, Sect. 6 presents concluding remarks and potential extensions of this work for future research.

2 Related Work

This section presents a review of studies that served as inspiration and reference for the development of the present study, focusing primarily on the control and maintenance of formations of multiple distributed agents aiming to surround a target without prior knowledge of its position or speed. For example, the work presented in [7] addresses the control of circular formation of multiple agents around a moving target with unknown and variable speed. The agents use local

sensors to capture information related to the target and their neighbors, considering connectivity constraints, collision avoidance, and uniformly spaced formation. The proposed controller employs the universal Lyapunov barrier function to ensure connectivity stability, formation maintenance, collision prevention, and operation under sensor limitations.

The study presented in [8] proposes a distributed algorithm for the formation of barriers by mobile agents around hazardous areas, using only local information and without prior knowledge of the boundaries of the risk region. The algorithm takes into account the detection and communication limitations of the agents, which operate within a specific radius, guiding the robots to form initial barriers at the edge of the hazardous area, followed by additional barriers at predefined distances. The process is divided into two main stages: border search and barrier formation. During the border search stage, agents adapt their movement based on their initial position: outside the area, near the edge, or inside it. Strategies such as spiral movement, direct movement, or straight-line paths are used to efficiently locate the border. In the barrier formation stage, agents adjust their positions based on the number of neighbors detected within the communication radius. Collision avoidance forces are applied to ensure proper spacing between agents, forming a uniform and functional barrier.

The work by [9] presents a decentralized control algorithm for swarms of minimalist robots aimed at encapsulating multiple targets while avoiding collisions with obstacles, whether static, dynamic, or with other swarm robots. The algorithm is based on three main components. In the initial phase, robots perform a random walk, autonomously exploring the environment and avoiding obstacles when no target is detected. In the second phase, upon detecting a signal below the safety threshold, robots approach the target, adjusting their positions in a coordinated manner to encapsulate it while continuing to avoid collisions. The use of the Lyapunov theorem in defining control parameters ensures stability during the approach. In the third phase, robots monitor signals above the safety threshold to prevent imminent collisions, adjusting their trajectories to maintain a minimum safe distance. The algorithm also addresses challenges such as sensor noise, asynchronous execution, and variations in control parameters, demonstrating robustness and scalability in simulations.

The study presented in [10] addresses formation control in distributed robotic swarms, focusing on multiple formations where robots must maintain appropriate distances from each other while navigating the environment and avoiding collisions with obstacles and other groups of agents. A distributed algorithm combining quadratic programming and control barrier functions was developed for this purpose. The barrier functions meet system requirements such as formation maintenance, collision prevention, and unified navigation. Formation maintenance is ensured by a rigid graph that defines the desired distances between pairs of robots, while obstacle collision avoidance is achieved by modeling obstacles as circles, ensuring that robots maintain a safe distance from their boundaries. To prevent collisions between groups, each formation is represented by an ellipse, enabling specific strategies to maintain safe distances. Additionally, the

algorithm optimizes communication by using only one robot per formation to exchange information. Quadratic programming is employed to determine control parameters that comply with all imposed constraints.

3 Methodological Background

The method proposed in this study is inspired by algorithms that simulate the collective behavior of animals moving in groups in nature, such as fish schools and bee swarms. The algorithms that underpin the developed approach are *Flocking* [11] and *Particle Swarm Optimization* (PSO) [12]. These models serve as the foundation for creating a methodology aimed at containing pollutants in aquatic environments.

3.1 Flocking

The *Flocking* algorithm was developed to realistically simulate the behavior and movements of bird flocks in flight. In the context of the algorithm, birds are represented by agents called *boids* [11]. Through the application of three simple rules, collective behavior emerges, allowing the swarm members to remain separated (avoiding collisions), aligned (moving in the same direction), and cohesive (staying close to one another). Applying these rules to *boid* i results in a total force, which is the weighted combination of the three forces generated by each rule. The basic rules are called separation, alignment, and cohesion.

Separation. The separation rule aims to prevent collisions between *boids*, ensuring that each one maintains a minimum distance from its closest neighbors. This is achieved by calculating a repulsion force proportional to the inverse square of the distance between *boid* i and its neighbors in the neighborhood $\mathcal{N}_i$. Mathematically, this rule can be expressed by Eq. 1:

$$\mathbf{f}_{\text{sep}}^{(t)} = k_{\text{sep}} \sum_{j \in \mathcal{N}_i^{(t)}} \frac{\mathbf{p}_i^{(t)} - \mathbf{p}_j^{(t)}}{\|\mathbf{p}_i^{(t)} - \mathbf{p}_j^{(t)}\|^2}, \tag{1}$$

where k_{sep} controls the intensity of the tendency for each agent to move away from its neighbors, $\mathcal{N}_i^{(t)}$ represents the set of neighbors near *boid* i at iteration t, defined by a perception radius, $\mathbf{p}_i^{(t)}$ is the position of *boid* i, and $\mathbf{p}_j$ is the position of a neighbor $j \in \mathcal{N}_i^{(t)}$ at iteration t.

Alignment. The alignment rule adjusts the movement direction of *boid* i to match the average direction of its neighbors, promoting uniformity in the group's behavior. The alignment force is calculated as the difference between the average velocity of the neighbors and the velocity of *boid* i, as shown in Eq. 2:

$$\mathbf{f}_{\text{ali}}^{(t)} = \frac{k_{\text{ali}}}{|\mathcal{N}_i^{(t)}|} \sum_{j \in \mathcal{N}_i^{(t)}} \mathbf{v}_j^{(t)} - \mathbf{v}_i^{(t)}, \tag{2}$$

where k_{ali} represents the tendency of each *boid* to align its velocity (direction and magnitude) with its neighbors, $|\mathcal{N}_i^{(t)}|$ is the number of neighbors of *boid* i, $\mathbf{v}_j^{(t)}$ is the velocity of a neighbor j, and $\mathbf{v}_i^{(t)}$ is the current velocity of *boid* i at iteration t.

Cohesion. The cohesion rule encourages each *boid* to move towards the center of mass of its neighbors, promoting group aggregation. This rule is described by Eq. 3:

$$\mathbf{f}_{\text{coh}}^{(t)} = \frac{k_{\text{coh}}}{|\mathcal{N}_i^{(t)}|} \sum_{j \in \mathcal{N}_i^{(t)}} \mathbf{p}_j^{(t)} - \mathbf{p}_i^{(t)}, \tag{3}$$

where k_{coh} defines the attraction force that each *boid* exerts to approach the group's center of mass. The term $\mathbf{p}_j^{(t)} - \mathbf{p}_i^{(t)}$ represents the vector pointing from *boid* i to the position of a neighbor j, and the average over all neighbors provides the direction to the local center of mass at iteration t.

Model Extensions. In addition to the basic rules, the *Flocking* model can be extended to incorporate new behaviors into the group. For example, it is possible to implement a rule that guides the group to chase a specific target or reach a predetermined position [13].

Chase. The chase behavior directs the *boid* to move directly toward a target point, **a**. This is achieved by applying a desired force toward the target and adjusting the *boid*'s velocity for consistent movement. This behavior can be mathematically described by Eq. 4:

$$\boldsymbol{f}_{\text{prsg}}^{(t)} = k_{\text{prsg}} \left(\frac{\boldsymbol{p}_a^{(t)} - \boldsymbol{p}_i^{(t)}}{\|\boldsymbol{p}_a^{(t)} - \boldsymbol{p}_i^{(t)}\|} \cdot \boldsymbol{v}_{max} - \boldsymbol{v}_i^{(t)} \right), \tag{4}$$

where k_{prsg} controls the intensity of the force generated during the chase, $\boldsymbol{p}_a^{(t)}$ and $\boldsymbol{p}_i^{(t)}$ are the positions of the target point and *boid* i at iteration t, respectively. The maximum velocity of the swarm's *boids* is denoted by $\boldsymbol{v}_{max}$, and $\boldsymbol{v}_i^{(t)}$ is the velocity of *boid* i at iteration t. Equation 4 models the force guiding the *boid* toward the target point. However, it does not account for deceleration as the *boid* approaches the target, which may lead to abrupt movements.

Approach. The approach behavior is an extension of the chase behavior. It introduces gradual deceleration as the *boid* nears the target point, $\mathbf{p}_a^t$. This approach prevents the *boid* from overshooting the target or arriving abruptly. The force calculation logic for *boid* i is detailed in Eq. 5.

$$f_{\text{aprx}}^{(t)} = \begin{cases} k_{\text{aprx}} \frac{\mathbf{p}_a^{(t)} - \mathbf{p}_i^{(t)}}{\|\mathbf{p}_a^{(t)} - \mathbf{p}_i^{(t)}\|} \cdot \mathbf{v}_{\max} \cdot \frac{\|\mathbf{p}_a^{(t)} - \mathbf{p}_i^{(t)}\|}{r}, & \text{if } \|\mathbf{p}_a^{(t)} - \mathbf{p}_i^{(t)}\| < r \\ k_{\text{aprx}} \frac{\mathbf{p}_a^{(t)} - \mathbf{p}_i^{(t)}}{\|\mathbf{p}_a^{(t)} - \mathbf{p}_i^{(t)}\|} \cdot \mathbf{v}_{\max}, & \text{if } \|\mathbf{p}_a^{(t)} - \mathbf{p}_i^{(t)}\| \geq r \end{cases} \tag{5}$$

where k_{aprx} defines how smoothly the progressive deceleration occurs, and the parameter r sets the deceleration radius. When the *boid* is at a distance $\|\mathbf{p}_a^{(t)} - \mathbf{p}_i^{(t)}\|$ less than r, it decelerates proportionally to the remaining distance, ensuring it reaches the target in a controlled manner without overshooting.

3.2 Particle Swarm Optimization

Particle Swarm Optimization is a metaheuristic inspired by the behavior of bird flocks, enabling the search for optimal values of a continuous objective function across the entire real domain. This search considers a d-dimensional space, where each variable has minimum and maximum limits. In PSO, a set of particles exchanges information among themselves, combining each particle's individual experience with the collective knowledge of the swarm to find the best global solution to the objective function [12]. The update dynamics of the particles, considering a limited field of view for the swarm members, is mathematically described by Eq. 6:

$$\mathbf{v}_i^{(t+1)} = k_{\text{pso}} \left(w \cdot \mathbf{v}_i^{(t)} + c_1 \cdot r_1^{(t)} \cdot \left(\mathbf{b}_i^{(t)} - \mathbf{p}_i^{(t)} \right) + c_2 \cdot r_2^{(t)} \cdot \left(\mathbf{g}_\alpha^{(t)} - \mathbf{p}_i^{(t)} \right) \right), \tag{6}$$

where k_{pso} is a constant used to enhance the result of the PSO algorithm, $\mathbf{p}_i^{(t)}$ and $\mathbf{v}_i^{(t)}$ are, respectively, the position and velocity vectors of particle i at iteration t, $\mathbf{b}_i^{(t)}$ represents the best position found by particle i in its trajectory, and $\mathbf{g}_\alpha^{(t)}$ corresponds to the best position discovered by its neighborhood. The term w is the inertia weight, while c_1 and c_2 are the cognitive and social coefficients, responsible for balancing local exploration and global exploitation. Finally, $r_1^{(t)}$ and $r_2^{(t)}$ are random numbers uniformly distributed in $[0, 1]$.

At each iteration, $\mathbf{g}_\alpha^{(t)}$ is updated if $f(\mathbf{p}_i^{(t)}) > f(\mathbf{g}_\alpha^{(t)})$, meaning that if particle i finds a solution better than the global best solution, the global best is updated. Similarly, the personal best position $\mathbf{b}_i^{(t)}$ is updated if $f(\mathbf{p}_i^{(t)}) > f(\mathbf{b}_i^{(t)})$.

4 Proposed Method

The method proposed in this work is based on the combined application of the separation, alignment, and cohesion rules, along with the behaviors of chasing and approaching a target point. Additionally, new rules were developed to enable the emergence of behaviors within the swarm for detecting and containing pollutants.

To understand how the developed rules operate, it is essential first to formalize the conceptual model of a *boid*, the fundamental element of the swarm. All members of the swarm share identical characteristics with the standard *boid*, serving as the starting point for defining and implementing the proposed rules. Figure 1 presents the conceptual representation of a *boid*.

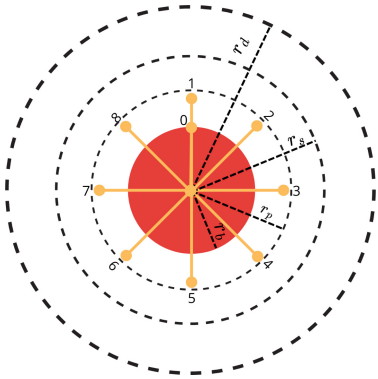


Fig. 1. Conceptual representation of a *boid*.

As shown in Fig. 1, the *boid* features four distinct radius: r_d, which defines the perception radius; r_s, corresponding to the safety radius to avoid collisions; r_p, representing the length of the arms equipped with probes at their ends; and r_b, referring to the size of the *boid*'s body.

The *boid* model includes nine probes to measure pollutant concentrations within its operating region. Eight of these probes are arranged around the body, as illustrated in Fig. 1, and are numbered from 1 to 8. There is also a central probe, identified as 0, capable of determining whether the *boid* is positioned over a contaminated area. The *boid* possesses a limited internal memory but is capable of storing relevant information from its local experience. This memory records the detected border position, denoted by $\boldsymbol{p}_i^{(e)}$, and the time when the *boid* encountered the pollutant's edge, as well as attributes required for PSO, such as the local optimum and its neighborhood's best.

With the conceptual model of the *boids* properly established, the algorithm for pollutant detection and containment was developed. This operation involves deploying a swarm of identical agents into the water, with prior knowledge of the pollutant's source. If the pollutant is merely floating in the ocean, for instance, this information consists of an approximate position of its center, indicated by $\boldsymbol{p}^{(\mathcal{S})}$. However, the agents have no knowledge of the shape or dimensions of the pollutant patch's borders, which are less dense than water. The agents' objective is to locate the boundary of this patch and encapsulate it, preventing its dispersion. Algorithm 1 details the procedure used to achieve this goal.

The algorithm begins by defining the fundamental parameters essential for the operation's effectiveness. The number of *boids* required to contain the pollu-

Algorithm 1. Detection and Containment of a Pollutant in Water.

Require: N_b, η, r_d, r_s, $p^{(\mathcal{S})}$, ξ, δ
Require: k_{sep}, k_{ali}, k_{coh}, k_{pso}, k_{prsg}, k_{a}, w, c_1, c_2
Ensure: $\Omega = 1$
$B \leftarrow$ initialize swarm with N_b; $f \leftarrow 0$; $\Omega \leftarrow 0$
while $\Omega = 0$ **do**
  for all $i \in B$ **do**
    $\boldsymbol{f} \leftarrow \mathbf{0}$
    if $\boldsymbol{p}_i^{(e)} \neq \emptyset$ **then**
      Check if *boid* i is in a dense region
      $\boldsymbol{p}_i^{(e)} \leftarrow$ Adjust position $\boldsymbol{p}_i^{(e)}$ of *boid* i using r_s, r_d and δ
      $\boldsymbol{f} \leftarrow$ Compute Eq. 5 for *boid* i using $\boldsymbol{p}_i^{(e)}$ as $\boldsymbol{p}_a$ and weight k_{aprx}
    else
      $\mathcal{N} \leftarrow$ Get neighborhood of i inside radius r_s
      $\mathcal{N}' \leftarrow \mathcal{N}' \cup \{p_k^{(e)} \mid p_k^{(e)} \neq \emptyset, k \in \mathcal{N}\}$
      if $|\mathcal{N}'| = 0$ **then**
        $\boldsymbol{f} \leftarrow \boldsymbol{f}+$ Compute Eq. 4 for *boid* i using $p^{(\mathcal{S})}$ as $\boldsymbol{p}_a$ and weight k_{prsg}
      end if
      $\beta \leftarrow$ Detect border using *boid* i with threshold η and radius r_s
      if $\beta = 1$ **then**
        $\boldsymbol{p}_i^{(e)} \leftarrow \boldsymbol{p}_i$
      end if
      $\boldsymbol{f} \leftarrow \boldsymbol{f}+$ Compute Flocking forces for *boid* i using k_{sep}, k_{ali}, k_{coh}
      $\boldsymbol{f} \leftarrow \boldsymbol{f}+$ Compute PSO force for *boid* i using w, c_1, c_2 and weight k_{pso}
      $\boldsymbol{f} \leftarrow \boldsymbol{f}+$ Compute polluted area avoidance force for *boid* i with η, r_s, k_{a}
    end if
    Move *boid* i using computed force $\boldsymbol{f}$
  end for
end while

tant is represented by N_b, while r_s corresponds to the safety radius, determining the minimum distance each *boid* must maintain from its immediate neighbor when positioning itself at the edge of the polluted region to avoid collisions. The parameter r_d refers to the perception radius of each *boid*, which defines the maximum distance for sending and receiving messages. The concentration threshold required to classify a region as highly polluted is indicated by η.

The collective behavior of the *boids* is regulated by a set of constants that influence their movement dynamics. The constants k_{sep}, k_{ali}, and k_{coh} determine the weights of the separation, alignment, and cohesion forces, respectively.

The term k_{pso} regulates the influence of the PSO Algorithm, while k_{prsg} controls the pursuit of the pollutant source, and k_{a} modulates the evasion of highly contaminated areas. Additionally, the parameters w, c_1, and c_2 represent the inertia coefficient and the weights associated with local and global knowledge in the PSO, respectively. The algorithm also employs a factor β, used to indicate whether the *boid* has already identified the position of the pollutant's edge, allowing efficient coordination of the collective formation.

After defining the initial parameters, the swarm is deployed, and the boolean variable Ω is introduced to indicate the complete formation of the barrier around the pollutant. The barrier is considered stabilized when the standard deviation and variance among the agents within its structure are less than or equal to ξ. While the pollutant is not fully contained, the algorithm remains in execution.

Each *boid* checks its internal memory to see if the position of the pollutant's edge has already been recorded. This verification is essential to prevent the agent from altering its previously identified position due to environmental influences. Next, it analyzes the validity of its position at the edge. If this edge has already been determined, the *boid* evaluates the number of agents within the radius r_s. If there is only one neighbor, it requests information from that neighbor regarding the total number of agents within r_s. If more than two agents are identified, *boid* i moves away from the edge, as its presence would compromise the uniformity of the pollutant containment barrier.

If this condition is not met, *boid* i recalculates its position to maintain an equidistant spacing from the nearest neighbor that has also identified the edge. This adjustment is performed according to the procedures described in Algorithm 2, ensuring the homogeneous distribution of agents and the stability of the barrier.

Algorithm 2. Adjust Position of *boid* i on the edge.

Require: r_d, $p_i^{(e)}$, r_s, δ
Ensure: $p_i^{(e)'}$
$\alpha \leftarrow 0.98 r_s$; $\beta \leftarrow 1.02 r_s$; $\mathcal{B} \leftarrow \emptyset$
$\mathcal{N} \leftarrow$ Get neighborhood of *boid* i inside radius r_d;
$\mathcal{N}' \leftarrow \mathcal{N}' \cup \{p_k^{(e)} \mid p_k^{(e)} \neq \emptyset,\ k \in \mathcal{N}\}$; $\mathcal{N}' \leftarrow$ Sort $\mathcal{N}'$ by distance to *boid* i
$\mathcal{B} \leftarrow \mathcal{B} \cup \{\boldsymbol{p}_k^{(e)} \mid \alpha \leq \|\boldsymbol{p}_k^{(e)} - \boldsymbol{p}_i^{(e)}\| \leq \beta,\ k \in \mathcal{N}'\}$
if $|\mathcal{N}'| = 0$ **then**
    $p_i^{(e)'} \leftarrow p_i^{(e)}$
else
    $j \leftarrow \mathcal{N}'[0]$
    if $|\mathcal{B}| = 1 \wedge t_i \geq \delta$ **then**
        $k \leftarrow \mathcal{N}'[1]$
        if $\|p_k^{(e)} - p_j^{(e)}\| < \|p_k^{(e)} - p_i^{(e)}\|$ **then**
            $\boldsymbol{p} = \boldsymbol{p}_k^{(e)}$
        else
            $\boldsymbol{p} = \boldsymbol{p}_j^{(e)}$
        end if
    else
        $\boldsymbol{p} \leftarrow \boldsymbol{p}_j^{(e)}$
    end if
end if
$\boldsymbol{p}_i^{(e)'} \leftarrow \boldsymbol{p} + r_s \cdot \frac{\boldsymbol{p}_i^{(e)} - \boldsymbol{p}}{\|\boldsymbol{p}_i^{(e)} - \boldsymbol{p}\|}$

Algorithm 2 performs a progressive adjustment of the edge position detected by *boid* i, ensuring a uniform distribution among the agents that have already identified this region of the pollutant. This adjustment is based on a vector operation that normalizes and scales the direction vector between the initial position of the edge detected by *boid* i and the position previously identified by its nearest neighbor. If *boid* i has a neighbor within a closed interval around r_s and has identified the edge for a time δ, it will adjust its position based on its second closest neighbor, *boid* k, located on the opposite side from i's nearest neighbor. To maintain the formation in Algorithm 1, each *boid* must remain at the position where it initially identified the edge. If this position is adjusted by Algorithm 2, the agent will consider the new position as its reference. For this purpose, Eq. 5 is applied, promoting the agent's gradual return to the recorded position whenever there is a deviation, ensuring the stability and cohesion of the containment barrier throughout the process.

During the execution of Algorithm 1, if the *boid* has not yet identified the edge position, the search phase begins. The first step is to guide the *boids* to move towards the pollutant source, represented by $\boldsymbol{p}^{(\mathcal{S})}$. This behavior aims to provide an initial indication of the direction to be followed so that the agents can reach their objective. However, once more than one *boid* in the neighborhood detects the edge, this tendency is deactivated. Next, the set of probes embedded in the *boid*'s body is read, and the data is stored in the variable S_i. With this information, it is verified whether the agent is at the pollutant's edge. The verification process follows the procedures defined in Algorithm 3, ensuring correct edge identification and allowing the containment strategy to proceed.

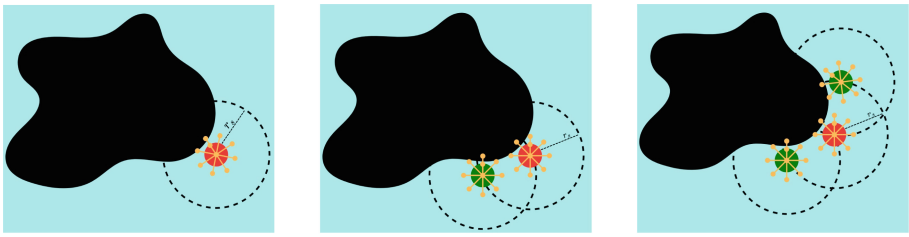
Algorithm 3. Detect the edge for *boid* i

Require: η, r_s
Ensure: β
1: $N_S^P \leftarrow$ Count *boid* i probes above η; $S_i^b \leftarrow$ Check if body probe values are below η
2: $\mathcal{N} \leftarrow$ Get neighborhood of *boid* i within r_s
3: **if** $N_S^P \geq 1 \wedge S_i^b = 0 \wedge |\mathcal{N}| < 2$ **then**
4: $\quad \beta \leftarrow 1$
5: **else**
6: $\quad \beta \leftarrow 0$
7: **end if**

Algorithm 3 is designed to identify whether the *boid* is at the boundary between regions of high and low pollutant concentration. Using the provided parameters, it verifies the values measured by the probes, counting the number of probes that register values above η, represented by N_S^P. It then checks if the probe located on the *boid*'s body, S_i^b, indicates a high concentration of pollutants. The set of neighbors within the safety radius r_s is also obtained.

The determination of whether the *boid* is at the edge is evaluated based on three criteria, with the result indicated by the variable β, which takes the value 1 if the *boid* is at the edge. First, N_S^P must be greater than or equal to

one, indicating that at least one of the probes detects a high concentration of pollutants. Second, S_i^b must not register a high concentration, ensuring that the body of the *boid* is not within the polluted region. Finally, the number of neighbors within the safety radius must be less than two, ensuring that the *boid* is not occupying a position already taken in the formation of the barrier.



(a) Situation 1 (b) Situation 2 (c) Situation 3

Fig. 2. Illustration of the application of the edge detection rule.

The *boid* detects the edge when at least one of its probes enters the pollutant region while its body probe remains outside. An additional condition is required to validate the position where the *boid* identifies the edge. Figure 2 illustrates how this condition functions. In the first scenario, illustrated in Fig. 2a, the *boid* detects the edge without any neighboring *boids* within radius r_s, making the position valid. In the second scenario, as shown in Fig. 2b, the *boid* detects the edge with one neighbor within radius r_s, which is still considered valid. However, in the third scenario, presented in Fig. 2c, the *boid* detects the edge but has two neighbors within radius r_s, rendering the position invalid as it is already occupied in the formed barrier.

In Algorithm 1, if the *boid* determines the edge position during the iteration (i.e., if $\beta = 1$), it records this position in its memory as its current location. Then, the rules associated with *Flocking* [11] are applied, starting a verification to determine whether the *boid* is in a region with other agents that have already identified the edge. If so, the *boid* considers only the edge agents in the calculation of alignment and cohesion rules, directing itself towards those that have already detected the edge, thus increasing its chances of identification. However, all agents, regardless of their position, are included in the calculation of the separation rule.

If the *boid* is in a region with a low concentration of agents that have identified the edge, all neighbors within r_d are considered in the calculation of the alignment and cohesion rules. After these verifications, the sum of the separation, alignment, and cohesion forces is applied, ensuring the maintenance of the swarm's collective behavior and the stability of the formation over time.

After applying the *Flocking* rules [11] in Algorithm 1, the PSO Algorithm [12], described in Eq. 6, is employed to guide the *boids* towards the edge of the pollutant. In this process, the highest concentration value detected in the

neighborhood is considered the global optimum. This information is obtained by querying the neighbors within radius r_d, and only these agents are considered in the evaluation. Moreover, *boids* that have already identified the edge position are automatically considered to hold the best global position in the neighborhood. Thus, the swarm tends to move toward the edge, ensuring that the containment of the pollutant is progressively reinforced by the collective dynamics of the agents.

While navigating the region in search of the pollutant edge, there is a possibility that the *boid* may enter an area with a high concentration of pollutants, potentially compromising both its operation and efficiency. To prevent this, the algorithm implements mechanisms that block the *boid*'s advancement under such conditions. These mechanisms are detailed in Algorithm 4, ensuring the *boid*'s safety and efficiency, allowing it to operate robustly and consistently in the environment.

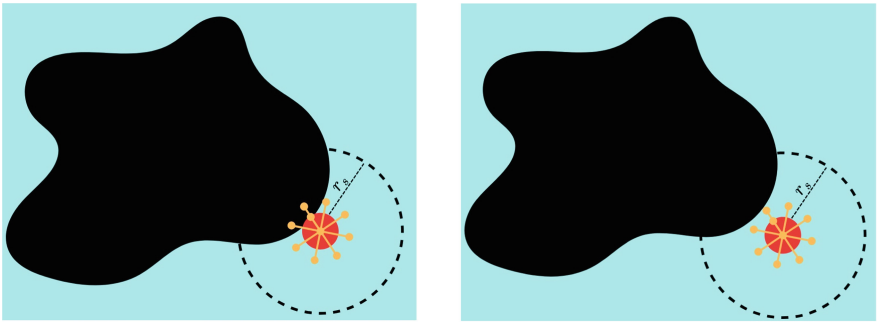
Algorithm 4. Prevent *boid* i from entering the polluted region

Require: k_a, η, r_s
Ensure: $\boldsymbol{f}$
1: $\boldsymbol{f} \leftarrow \mathbf{0}$; $S_i \leftarrow$ Read probe values of *boid* i
2: $N_S^P \leftarrow$ Count *boid* i probes above η; $S_i^b \leftarrow$ check if body probe values are below η
3: **if** $N_S^P > 2 \vee S_i^b = 1$ **then**
4: $\quad \boldsymbol{p}_i^{(s)} \leftarrow \{\text{Position of } s \mid s \in S_i \wedge s \leq \eta\}$
5: $\quad \boldsymbol{f} \leftarrow$ Compute Eq. 5 for *boid* i using $\boldsymbol{p}_i^{(s)}$ as $\boldsymbol{p}_a$ and weight k_a
6: **end if**

The rule presented in Algorithm 4 performs an initial check of the probes of *boid* i, obtaining the values of N_S^P and S_i^b, as described in Algorithm 3. It then checks if the number of probes with values above η is greater than two, indicating that due to inertia, the *boid* will enter the polluted zone, or if the probe located on the *boid*'s body, S_i^b, is detecting pollutants. If neither condition is met, no action is taken. If at least one condition is true, the algorithm identifies the probe that is not detecting a concentration above η, as illustrated in Fig. 3. Based on this information, the *boid* calculates an approach to the position associated with this probe using Eq. 5. The force resulting from the approach equation is then used as the output of Algorithm 4.

Continuing with Algorithm 1, *boid* i is moved according to the direction of the resulting vector from the applied forces. With the complete update of the swarm, the stopping condition is evaluated based on the agents that have already determined the edge position.

Each *boid* on the edge transmits the distance to its closest neighbors to its adjacent neighbor, propagating this information throughout the formation until all agents on the edge are aware of these distances. After this step, the standard deviation and variance of these measurements are calculated. If the values of these metrics are less than or equal to ξ, it is considered that the algorithm



(a) *Boid* enters polluted zone. (b) *Boid* backs out of the polluted zone.

Fig. 3. Illustration of the rule that prevents the *boid* entry into polluted zone.

has reached convergence, ensuring the stabilization of the barrier around the pollutant.

5 Results

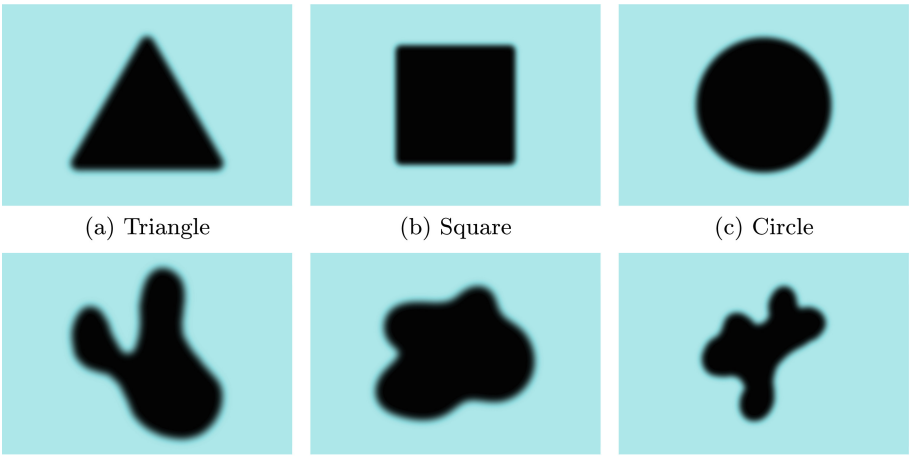
In this section, we present the results obtained from the execution of the proposed approach. The experiments were conducted using the CPU of a computer equipped with an Intel Core i5-10400F processor, 2.9 GHz, and 16 GB of RAM.

To simulate the shape of the pollutant patch on the water surface, different shapes are generated. Initially, simple geometric shapes such as triangles, squares, and circles are used. Subsequently, additional tests are conducted using abstract shapes to showcase the applicability of the proposed approach to real-world scenarios. Figure 4 illustrates the synthetic shapes used in the experiments.

In Fig. 4, the pollutant patch is represented by the color black, while its surrounding blue area symbolizes the ocean. The sensing performed by the *boids* is based on extracting the RGB values from the region where their probes are positioned, with the color readings normalized on a scale from 0 to 1.

The shapes presented in Fig. 4 exhibit a slight gradient in the color tone representing the patch, allowing the agents to detect the edge through a threshold defined in Algorithm 1 as η. This approach is essential because, in the physical model, agents use sensors capable of measuring continuous pollutant concentrations. Without the color gradient, detection would occur in a binary manner, which would not accurately reflect the reality of the system. Therefore, the method employed enables a gradual transition in edge perception, making detection closer to the expected behavior in a physical environment.

Once the shapes to be used in the experiments were defined, the parameters for Algorithm 1 were set to ensure the best behavior for the swarm and the fastest convergence, allowing the algorithm to be evaluated under different conditions. The parameters used in the simulations are presented in Table 1.



(a) Triangle (b) Square (c) Circle

(d) Abstract$_1$ (e) Abstract$_2$ (f) Abstract$_3$

Fig. 4. Set of synthetic forms created for experimental validation.

Table 1. Parameters setting.

Swarm setup		Boids parameters		PSO parameters		Flocking parameters	
N_b	96	r_b	10	w	2.9	k_{sep}	7.5
$\boldsymbol{v}_{\max}$	2	r_d	80	c_1	1.8	k_{ali}	1.7
η	0.22	r_s	$0.35r_d$	c_2	4	k_{coh}	1.7
$\boldsymbol{p}^{(\mathcal{S})}$	(600,500)	r_p	$0.2r_d$	k_{pso}	1.5	k_{prsg}	0.4
ξ	1.5	δ	5	-	-	k_{aprx}	3.0
–	–	–	–	–	–	k_{a}	12.5

In the experiments, the performance of the algorithm was evaluated based on the number of iterations required for convergence and the execution time. To ensure the reproducibility of the results, each experiment was executed 150 times, enabling the extraction and analysis of descriptive statistics for these metrics through tables and graphs.

Additionally, different initial configurations were tested by varying the launch positions of the agents, which could be distributed from any of the four corners of the figure. This strategy allowed for a more comprehensive evaluation of the algorithm's dynamics, assessing its ability to efficiently and distributively coordinate the identification and containment of the pollutant among the agents.

In the figures and tables used for the results analysis, the synthetic shapes considered are represented by *Shape*, while the number of distinct launch sites is indicated by $\#LS$. When two launch sites are used, the notation $2'$ denotes a variation in launch site location. The total number of iterations executed by the algorithm is indicated by $\#IT$, and the execution time is represented by ET.

The first experiment conducted to evaluate the effectiveness of the developed method involved applying the algorithm to known geometric shapes previously presented in Fig. 4. This approach allowed testing the agents ability to detect and contain well-defined edges, providing a controlled environment for performance analysis.

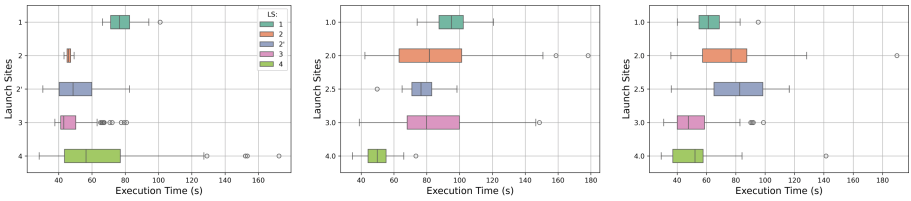
Table 2 presents the mean, standard deviation, minimum, and maximum values for the total number of iterations and execution time across different geometric shapes and launch sites. The table analysis corroborates the hypothesis that the initial positions of the agents and the shape of the edge directly influence the algorithm's performance, resulting in significant variations in execution time and the number of iterations. The analysis highlights that the square is the most challenging shape to be contained, exhibiting the highest average execution time ($94.84 \pm 9.39\ s$) and the largest average number of iterations (5690.31 ± 563.48).

Table 2. Iterations and execution time for different geometric shapes across launch sites.

Shape	#LS	#IT				ET (s)			
		Mean	Std	Min	Max	Mean	Std	Min	Max
Triangle	1	4610.68	396.33	3977	6055	76.84	6.61	66.28	100.92
	2	2765.45	76.66	2591	2956	46.09	1.28	43.18	49.27
	2'	3015.39	730.52	1828	4955	50.26	12.18	30.47	82.58
	3	2811.23	525.8	2262	4845	46.85	8.76	37.7	80.75
	4	3762.57	1508.03	1695	10344	62.71	25.13	28.25	172.4
Square	1	5690.31	563.48	4451	7238	94.84	9.39	74.18	120.63
	2	5100.58	1560.09	2540	10696	85.01	26.0	42.33	178.27
	2'	4633.79	484.75	2987	5906	77.23	8.08	49.78	98.43
	3	5023.41	1559.17	2332	8918	83.72	25.99	38.87	148.63
	4	3003.37	450.48	2080	4401	50.06	7.51	34.67	73.35
Circle	1	3693.31	546.45	2406	5714	61.56	9.11	40.1	95.23
	2	4540.86	1315.5	2146	11388	75.68	21.93	35.77	189.8
	2'	4879.49	1198.71	2162	6986	81.32	19.98	36.03	116.43
	3	3035.83	803.54	1853	5929	50.6	13.39	30.88	98.82
	4	2944.4	814.09	1751	8490	49.07	13.57	29.18	141.5

This complexity arises from the difficulty of evenly dispersing agents around its sharp corners, leading to longer stabilization periods. In contrast, the triangle demonstrated the shortest average execution time ($46.85 \pm 8.76\ s$) and a reduced number of iterations (2811.23 ± 525.8), indicating a more efficient containment process. The circle, while presenting intermediate performance, displayed good uniformity in agent distribution, with execution times varying from $49.07 \pm 13.57\ s$ to $81.32 \pm 19.98\ s$, depending on the launch site.

To further illustrate these findings, Fig. 5 presents boxplots of execution times for the different geometric shapes across various launch sites. The boxplots visually depict the distribution and variability of execution times, highlighting the impact of shape complexity and initial agent positions. Notably, the triangle consistently shows the shortest execution times, while the square reveals the largest spread, reinforcing its higher computational demand.



(a) Triangle (b) Square (c) Circle

Fig. 5. Boxplot of execution time for different geometric shapes across launch sites.

The presence of outliers in the boxplots suggests that the swarm's dynamics, influenced by the *Flocking* rules [11], occasionally led to random walks, impacting performance. Additionally, the initial distribution of agents played a crucial role in determining execution time, with varying launch points leading to notable differences in containment efficiency. This behavior is further visualized in the video available at: https://youtu.be/RFz8T2-FrYA, where the swarm dynamics and the algorithm's adaptation to distinct geometries are demonstrated.

Following the experiments with geometric shapes, the algorithm's ability to detect and contain a patch without a predefined shape was evaluated. For this purpose, the same parameters from the previous experiments were maintained, with only the shape configurations being altered. The new shapes used are represented in Fig. 4d–4f.

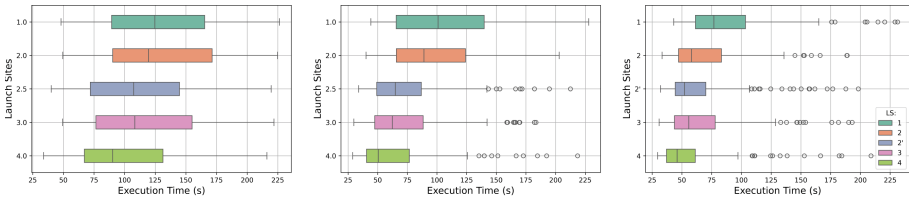
Table 3 presents the statistical data for the number of iterations and execution times across the tested abstract shapes and launch sites. The data confirm the hypothesis that the structural complexity of the shape and the variation in launch locations significantly affect the algorithm's performance. The containment of shape Abstract_1 shows the highest average execution times (104.63 s to 129.76 s) and iteration counts, while Abstract_3 demonstrates greater computational efficiency, with execution times ranging from 56.91 s to 88.55 s and fewer required iterations.

The data from Table 3 emphasize the influence of structural complexity and launch strategies on the containment process. The abstract shape Abstract_1 consistently required higher computational effort, while Abstract_3 exhibited a significant reduction in both execution time and iterations needed to stabilize the containment barrier. Additionally, the variation in launch sites directly impacted performance, with LS 4 consistently producing the best results across all shapes, significantly optimizing the algorithm's efficiency.

Table 3. Iterations and execution time for different abstract shapes across launch sites.

Shape	#LS	#IT				ET (s)			
		Mean	Std	Min	Max	Mean	Std	Min	Max
$Abstract_1$	1	7685.52	2732.61	2886	13585	128.09	45.54	48.1	226.42
	2	7785.65	3037.78	2972	13491	129.76	50.63	49.53	224.85
	2'	6853.44	2938.06	2405	13179	114.22	48.97	40.08	219.65
	3	7046.65	2844.43	2963	13317	117.44	47.41	49.38	221.95
	4	6277.76	2757.76	2031	12968	104.63	45.96	33.85	216.13
$Abstract_2$	1	6333.29	2694.35	2652	13677	105.55	44.91	44.2	227.95
	2	5942.18	2477.97	2411	12184	99.04	41.3	40.18	203.07
	2'	4464.37	2144.65	2031	12754	74.41	35.74	33.85	212.57
	3	4439.11	2183.51	1794	11011	73.99	36.39	29.9	183.52
	4	3831.38	2122.95	1738	13114	63.86	35.38	28.97	218.57
$Abstract_3$	1	5312.95	2353.21	2588	13880	88.55	39.22	43.13	231.33
	2	4145.83	1861.49	2001	11334	69.1	31.02	33.35	188.9
	2'	3921.22	2043.59	1920	11894	65.35	34.06	32.0	198.23
	3	4063.69	2132.92	1851	11583	67.73	35.55	30.85	193.05
	4	3414.47	1927.68	1775	12533	56.91	32.13	29.58	208.88

To further visualize these findings, Fig. 6 presents boxplots illustrating the distribution of execution times across different abstract shapes and launch sites. The figure highlights the spread and variability in execution times, reinforcing the observations derived from the statistical analysis. In particular, $Abstract_1$ displays a wider dispersion due to hard-to-reach regions, while $Abstract_3$ achieves a more consistent performance.



(a) $Abstract_1$ (b) $Abstract_2$ (c) $Abstract_3$

Fig. 6. Boxplot of execution time for different abstract shapes across launch sites.

The boxplots also reveal that in all scenarios, launching agents from four distinct points (LS 4) yielded the best containment results, evidenced by lower median execution times and reduced variability. This behavior is further validated in the video available at https://youtu.be/jLiOLX1hfOM, where the

dynamics of agent distribution and the impact of varying launch strategies on containment efficiency are clearly demonstrated.

6 Conclusions

This study presented a swarm intelligence-based approach for the containment of low-density water pollutants, inspired by collective behaviors observed in nature, such as schools of fish and swarms of bees. The proposed methodology integrates the concepts of *Flocking* and Particle Swarm Optimization, combined with specific rules for detecting and dynamically forming barriers, providing a distributed, scalable, and efficient solution. The simulation results demonstrated that the developed method is capable of detecting and containing pollutant patches with varying geometries and levels of complexity, influenced by the initial distribution of agents and the shape of the patch. More regular shapes favored reduced containment times, while irregular geometries required a higher number of iterations. Among the main contributions, the application of swarm intelligence in mitigating environmental impacts stands out, offering an efficient alternative to traditional physical barriers.

For future work, we intend to reduce the algorithm's execution time, validate the method in robotic simulations, and implement parallel execution, allowing each *boid* to operate independently and in a decentralized manner. Furthermore, the incorporation of physical variables such as wind, tides, and currents will be essential to evaluate the algorithm's performance under dynamic environmental conditions, ensuring greater robustness and applicability in real-world scenarios. Although the current work outlines these objectives, the analysis of the simulation results could be enhanced through the inclusion of quantitative performance metrics, such as containment time, covered area, the workload distributed among agents, or comparative evaluations with existing approaches. This would allow a more rigorous assessment of the method's effectiveness and support its validation in future studies.

References

1. Silva, D.C.P., dos Santos Melo, C., de Oliveira, A.B., dos Santos, N.M.M., Pinto, L.C.: Oil spill in the sea and toxic implications of exposure to petroleum chemical. *Revista Contexto Saúde***21**(44), 332–344 (2022)
2. Soares, M.O., Rabelo, E.F.: Severe ecological impacts caused by one of the worst orphan oil spills worldwide. Mar. Environ. Res. **187**, 105936 (2023)
3. Dhaka, A., Chattopadhyay, P.: A review on physical remediation techniques for treatment of marine oil spills. J. Environ. Manage. **288**, 112428 (2021)
4. Muttin, F.: Structural analysis of oil-spill containment booms in coastal and estuary waters. Appl. Ocean Res. **30**(2), 107–112 (2008)
5. Zahugi, E.M.H., Shanta, M.M., Prasad, T.V.: Oil spill cleaning up using swarm of robots. In: Meghanathan, N., Nagamalai, D., Chaki, N. (eds.) Advances in Computing and Information Technology. Advances in Intelligent Systems and Computing, vol. 178. Springer, Berlin (2013). https://doi.org/10.1007/978-3-642-31600-5_22

6. Kounte, M.R., Raghavendra Kashyap, T.M., Rahul, P., Ramyashree, M.K., Riya, J.K.: Oil spill detection and confrontation using instance segmentation and swarm intelligence. In: Mallick, P.K., Bhoi, A.K., Marques, G., Hugo C. de Albuquerque, V. (eds.) Cognitive Informatics and Soft Computing. AISC, vol. 1317, pp. 247–259. Springer, Singapore (2021). https://doi.org/10.1007/978-981-16-1056-1_20
7. Xiong, Y., Lu, K., Dai, S.-L.: Cooperative circular formation control design for enclosing a moving target with collision avoidance. In: 2022 41st Chinese Control Conference (CCC), pp. 4551–4556 (2022)
8. Shao, Y., et al.: Autobar: automatic barrier coverage formation for danger keep out applications in smart city. Sensors **23**(18) (2023)
9. Sinhmar, H., Gazit, H.K.: Decentralized control of minimalistic robotic swarms for guaranteed target encapsulation. In: 2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE (October 2022)
10. Aynala, A., Atman, M.W.S., Gusrialdi, A.: Communication-efficient formation maintenance for multi-robot system with a safety certificate. In: 2022 IEEE Conference on Control Technology and Applications (CCTA), pp. 608–613 (2022)
11. Reynolds, C.W.: Flocks, herds, and schools: a distributed behavioral model, pp. 273–282. Association for Computing Machinery, New York, USA (1998)
12. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: Proceedings of ICNN 1995 - International Conference on Neural Networks, vol. 4, pp. 1942–1948 (1995)
13. Reynolds, C.W., et al.: Steering behaviors for autonomous characters. In: Game Developers Conference, vol. 1999, pp. 763–782. Citeseer (1999)