Rescaling for Area Coverage in Swarm Robotics

Nadia Nedjah[1(✉)], Luiza de Macedo Mourelle[2], and Luneque Del Rio de Souza e Silva Junior[3]

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

[2] Department of Systems Engineering and Computation, State University of Rio de Janeiro, Rio de Janeiro, Brazil
ldmm@eng.uerj.br

[3] Departamento de Information Engineering, Federal University of the ABC, São Paulo, Brazil
luneque.junior@ufabc.edu.br

Abstract. Swarm robotics studies collective behaviors in systems composed of multiple robots. In this work, a message wave strategy is used in which the swarm operates as a network of robots that interact through wireless communication. A distributed message wave algorithm is designed for use in robot swarms and serves as a foundation for controlling different collective tasks. This work introduces a general methodology that provides a systematic approach to task execution in robotic swarms. Using this methodology, control strategies for the well-known area coverage problem are proposed, implemented, and evaluated through simulation, focusing on rescaling behaviors such as aggregation and dispersion. Experimental results show that the time required to complete these tasks depends on the swarm size, message propagation, and the specific characteristics of each task. The design and execution of these tasks serve as a proof of concept for the proposed strategy, demonstrating its generality and potential applicability to a wide range of other collective swarm behaviors.

Keywords: Swarm robotics · area coverage · aggregation · dispersion

1 Introduction

Swarm robotics is a multidisciplinary research field concerned with the study and development of systems composed of multiple robots capable of performing tasks collectively.

Collective task execution is a central topic in swarm robotics research. A task is defined as an operation performed by a robot, such as moving a given distance or maintaining motion for a specified time. Tasks become collective when their execution depends not only on the actions of an individual robot

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2027
O. Gervasi et al. (Eds.): ICCSA 2026 Workshops, LNCS 16761, pp. 3–21, 2027.
https://doi.org/10.1007/978-3-032-30527-5_1

but also on the actions of other robots in the swarm. In such cases, interaction among robots is required, which may occur through communication, via explicit information exchange or through sensing, whereby robots detect the presence of nearby swarm members.

Numerous collective tasks have been investigated in the literature [1,3,15]. However, many existing approaches focus on designing task-specific control software. As a result, each new behavior typically requires the development of a dedicated controller that accounts for both the task requirements and the robot hardware. While effective, this approach entails considerable design effort and limits the reuse and scalability of control strategies across different tasks.

The message wave strategy is task-independent and serves as a general foundation for implementing diverse collective behaviors [26,28]. It relies on two core processes: the propagation of messages from an initiating robot and the transmission of feedback messages in the reverse direction. Together, these processes enable coordinated task execution and synchronization, allowing subsequent tasks to begin only after the completion of previous ones.

The wave strategy is validated through the implementation of multiple tasks in a simulated swarm environment [10,27]. The simulation platform models physical phenomena such as gravity, friction, inertia, and collisions, while allowing customization of robot models. Experiments were conducted using a simulated version of the Kilobot [21], whose minimal communication capabilities are sufficient for message wave propagation. In particular, this work focuses on the design, implementation, and evaluation of rescaling collective behaviors for spatial area coverage, enabling aggregation and dispersion in robot swarms. These tasks serve as proof of concept for the proposed strategy, demonstrating its generality and applicability to a wide range of collective swarm behaviors.

Robot aggregation is required when a swarm that is initially dispersed needs to occupy a smaller region than its current spatial footprint. Similar to recruitment, aggregation can be employed as a subtask in several applications, including:

- Swarm collection: robots distributed throughout the environment during task execution must be gathered into a confined region so they can be retrieved by the user upon task completion.
- Object transportation: robots detect and grasp objects in the environment and collectively transport them to a designated location.
- Pursuit and capture: aggregation can be used to intercept a target by concentrating robots around it, thereby restricting its movement and preventing escape.
- Environmental transition: a moving group of robots may need to pass through a narrow passage or transition from a larger environment to a smaller one.

Dispersion exhibits the opposite behavior of aggregation: robots initially concentrated in a small area must spread out over the environment in which the swarm will operate. Dispersion-based subtasks can be applied in several scenarios:

- Initialization dispersion: the swarm is initially deployed near the user and subsequently disperses over a larger area. This behavior is particularly useful in search tasks, where robots must spread throughout the environment to locate regions with specific characteristics or to find objects.
- Coverage: in many applications, it is desirable for the swarm to occupy the largest possible area while maintaining communication connectivity, thereby avoiding fragmentation of the swarm topology.
- Localization support: once dispersed, some robots may remain stationary so that their positions can serve as reference points in a localization system.

This paper is organized into six sections. After the introduction, Sect. 2 presents the state of the art regarding the most widely studied and required collective tasks. Section 3 describes the general characteristics required of the swarm communication system to enable distributed algorithms, specifically information propagation using the wave strategy. Section 4 introduces rescaling tasks implemented using the message wave strategy, which enable aggregation and dispersion behaviors. Section 5, discusses the obtained performance results for several case-studies. Finally, Sect. 6 presents concluding remarks and outlines promising directions for future research.

2 Collective Tasks in Swarm Robotics

The analysis of survey and review works in swarm robotics provides valuable insight not only into how the field has evolved over the years, but also into how it has been interpreted by different researchers. Early studies highlighted the need to classify systems based on global characteristics, such as the architectural organization of the swarm, as proposed in [5]. Other authors have offered alternative perspectives, such as the work in [9], in which system characteristics are interdependent rather than treated in isolation.

In general, swarm tasks address three main aspects: (i) spatial organization among robots and/or between robots and the environment; (ii) robot motion within the environment, including search, object detection, and object transport; and (iii) collective decision-making processes, also known as consensus [3,15]. A wide range of problems and tasks has been investigated in swarm robotics.

One major class of swarm tasks concerns spatial organization, in which robots adjust their relative positions, either moving closer together or farther apart, leading to emergent global properties. Well-known examples include aggregation, dispersion, area coverage, pattern formation, and self-assembly. Aggregation, inspired by behaviors observed in insect societies [2], requires robots to converge toward a common region and has been widely studied due to its relevance to applications such as cooperative transport [13,14] and pattern formation [28]. Many aggregation strategies are biologically inspired, drawing from studies of bees [24] and cockroaches [8], where environmental cues combined with local interactions guide collective behavior. Dispersion represents the opposite behavior, in which robots spread over a larger area, often while maintaining connectivity. This behavior is closely related to the area coverage problem, which

is essential for applications such as environmental monitoring and distributed sensing [4]. More structured forms of spatial organization include pattern formation [7] and self-assembly, where robots physically connect to form composite structures, as explored in projects such as Symbrion [11].

Another important category of swarm tasks involves collective movement, in which the swarm as a whole moves from one region to another. This category includes exploration, foraging, collective navigation, and collective transport. Exploration of unknown environments is a fundamental robotics problem, often addressed using swarms to exploit parallelism and redundancy [6]. Foraging tasks, inspired by food-gathering behaviors in social insects, are widely regarded as benchmark problems in cooperative robotics [29], involving search, object identification, transport, and deposition [13,14]. Collective navigation, also known as flocking, refers to coordinated group motion and is inspired by biological systems [19,28]. Collective transport addresses the cooperative movement of objects that exceed the capabilities of individual robots and has been extensively studied using both communication-free and physics-based decentralized approaches, as well as methods relying on partial global information [14].

A third fundamental class of tasks involves collective decision-making, which is essential in decentralized swarm systems that lack global information. These tasks include consensus, task allocation, and localization. Consensus problems address how robots converge to a shared decision through local interactions [12]. Task allocation, also known as division of labor, concerns how robots distribute themselves among multiple tasks to achieve a desired global proportion, often using threshold-based or probabilistic methods [16,17]. Localization is another critical problem, where robots estimate their positions without external references such as GPS, typically relying on inter-robot measurements and distributed estimation techniques [22,23].

3 Distributed Algorithms for Swarm Robotics

Wave algorithms are a class of algorithms that manage message passing in distributed systems [26]. These algorithms are commonly used as a foundation for tasks that require an organized and reliable message transmission scheme. To be executed, wave algorithms require an architecture composed of devices capable of sending and receiving messages. The robotic system must exhibit the necessary characteristics for the execution of message waves:

- The swarm communication structure forms a network whose topology is defined by a graph $G = (V, A)$, where V is the set of robots and A is the set of communication channels between robots.
- The network topology does not change during the execution of the algorithm.
- The network is connected, meaning that every robot can transmit information, directly or indirectly, to any other robot in the swarm.
- Robots possess some initial knowledge, such as their own identifier or the identifiers of neighboring robots.

3.1 Information Propagation with Feedback

Wave algorithms can be employed to perform information propagation in distributed networks [25]. This operation is essential in systems that do not support broadcast communication, such as robot swarms that rely exclusively on local interactions.

During information propagation, each process completes its participation by forwarding messages to its neighbors. A process may confirm that its neighbors have received the information through acknowledgment messages; however, it has no direct confirmation regarding processes outside its local neighborhood. The Propagation of Information with Feedback (PIF) algorithm [25–27] extends basic propagation by providing an explicit event that indicates when the information has been received by all processes.

Synchronization can be achieved through successive executions of the PIF algorithm. In this case, propagation messages do not carry data, but instead signal that a local action a_0 may be executed. Once all processes have completed a_0, feedback messages are returned to the initiator, triggering the decision event. The initiator then starts a new wave, allowing each process to execute action a_1. Since the execution of a_1 occurs after the decision event, which is causally dependent on the completion of a_0 by all processes, global synchronization is guaranteed.

4 Proposed Model for Spatial Resizing Tasks

An important collective behavior that can be implemented using the message-wave strategy is swarm resizing or rescaling. This task requires a robot model equipped with sensing capabilities beyond those used in collective navigation. Specifically, each robot must be able to perceive the positions of all neighboring robots. Using this local spatial information, robots autonomously adjust their positions by increasing or decreasing their inter-robot distances. Depending on the selected control parameters, the swarm resizing process can lead to two emergent behaviors: aggregation or dispersion.

Generally, robot's model estimates inter-robot distances based on the intensity of the received communication signal. Accordingly, the simulated model associates distance measurements with inter-robot communication, thereby emulating the sensing mechanism available in the robot physical platform. Usually, this measurement provides information only about the distance between robots, without indicating their relative direction.

This task also requires neighborhood information to be collected. In this context, each robot uses the relative position of its parent to determine the speed and direction of its movement. To enable this behavior, the robot's model is modified so that each robot can observe the position of its corresponding parent robot during message reception. This modeling choice, however, ignores the positions of other neighboring robots, since such information is not required for the leader-follower control scheme. The modified model also incorporates a distance sensor that detects the position of the nearest obstacle. Moreover, the resizing

tasks reveal the need to detect the positions of all robots and obstacles within a defined neighborhood around each robot. This information is necessary to prevent collisions and to preserve connectivity within the swarm. This requirement differs from the coordinated motion task presented in [28], in which complete neighborhood information is not essential.

4.1 Agent's Flocking

The term flocking refers, in the context of artificial swarms, to the computational modeling of collective motion in groups of agents. This concept was originally introduced by Reynolds [18] for generating realistic animations of animal group movements.

Each individual in artificial flocking operates based on observations of its local neighborhood. An agent identifies nearby individuals and uses their positions and velocities to compute its own velocity. This process of (*i*) sensing, (*ii*) computation, and (*iii*) velocity update is continuously repeated during the flocking behavior.

The velocity of each individual (or agent) is defined by the contribution of different behavioral rules. Each rule corresponds to a specific group characteristic that should be reinforced or attenuated. The three fundamental rules are separation, alignment, and cohesion. The operation of these rules is illustrated in Fig. 1. In all cases, the dashed line represents the detection area surrounding the highlighted individual. As a result, this individual interacts with only three neighbors, while the positions of other agents do not influence its movement.

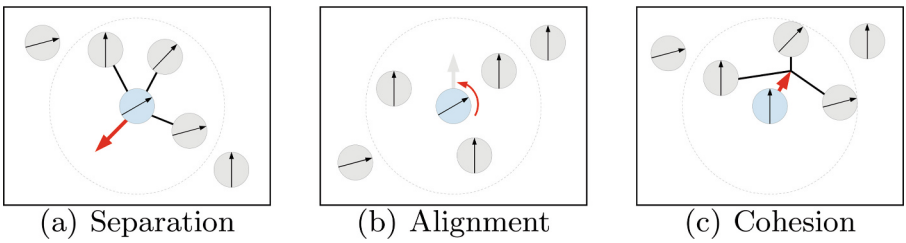


Fig. 1. Three basic flocking rules.

Separation is illustrated in Fig. 1(a). Under this rule, an individual moves away from its neighbors, preventing overcrowding and avoiding collisions. In the alignment rule shown in Fig. 1(b), the individual adjusts its direction in order to match the average heading of its neighbors. Finally, the cohesion rule depicted in Fig. 1(c) aims to prevent individuals from drifting away from the group. To achieve this, the computed movement is directed toward the average position of neighboring agents. In general, the flocking process is achieved through repeated cycles of neighborhood observation and position adjustment, as illustrated in

Algorithm 1. Each agent moves based on the observation of its own neighborhood. However, this movement alters the positions of all agents, meaning that at each sensing step, individuals perceive their neighbors at gradually changing locations. The contribution of each flocking rule is weighted by parameters w_1, w_2, and w_3 in Algorithm 1. The parameter w_0 represents the weight of the agent's current velocity relative to the newly computed velocity components. This factor models inertia; without it, abrupt velocity changes could occur due to fluctuations in the neighborhood configuration. TagImg

Algorithm 1. flocking of individuals.

1: Randomly initialize the individual's velocity;
2: **while** true **do**
3: Detect the neighborhood; Acquire neighborhood position information;
4: v_1 = separation velocity using neighborhood position information;
5: v_2 = alignment velocity using neighborhood position information;
6: v_3 = cohesion velocity using neighborhood position information;
7: velocity = $w_0 \times$ velocity + $(w_1 \times v_1)$ + $(w_2 \times v_2)$ + $(w_3 \times v_3)$;
8: position = position + velocity $\times \Delta t$;
9: **end while**

The tuning of the flocking parameters leads the group to exhibit different collective behaviors. For instance, varying w_0 can increase or decrease the overall speed of the group. Conversely, changing the balance between separation and cohesion, by varying w_1 and w_2, adjusts the spatial distribution of agents until an equilibrium condition is reached. The cohesion component causes agents to move closer together, while separation induces mutual repulsion. This balance between cohesion and separation forms the foundation of swarm resizing tasks in robot swarms.

4.2 Aggregation and Dispertion of Robots

Swarm resizing is a spatial organization behavior in which robots adjust their distances relative to their neighbors in order to achieve a desired collective property. Two resulting tasks are aggregation and dispersion. In aggregation, all robots move closer to one another, reducing the area of the environment occupied by the swarm. In dispersion, the opposite occurs: robots mutually repel each other, increasing the swarm's area of operation.

The flocking rules enable a swarm to exhibit different spatial behaviors. The aggregation and dispersion tasks presented in this chapter employ these rules to control the movement of the robots composing the swarm. In addition, swarm resizing based on flocking rules follows the message-wave strategy. Accordingly, the resizing process exhibits the following characteristics:

– A recruitment subtask is executed prior to the onset of movement. This establishes parent–child relationships among robots, which are required for message-based communication within the swarm [27].

- A message wave determines both the start and the termination of the resizing process.
- The movement strategy prevents parent robots and their respective children from moving beyond their communication range.
- During aggregation, a robot with two or more neighbors must avoid excessive distancing from one neighbor in order to approach another, as this could create robot-free regions within the swarm.
- Similarly, during dispersion, a robot must avoid approaching some neighbors while distancing itself from others. Instead, repulsion must occur in a balanced and mutual manner throughout the swarm.

During swarm resizing, robot motion is governed by flocking rules in conjunction with the detection system. Each robot observes its local neighborhood and computes its displacement accordingly. At the same time, the message-based communication system ensures that aggregation or dispersion is interrupted once a predefined stopping condition is reached.

The distributed algorithm governing swarm resizing consists of two components that operate in a coordinated manner. One component corresponds to robot motion based on the flocking rules, where each robot's displacement is determined by the relative positions of its neighboring robots. The second component is a message-wave algorithm that signals to swarm members the beginning and the termination of the resizing process. In general, the message-wave strategy employed for swarm resizing is identical to that used during alignment. An initial message wave is responsible for recruiting the robots and establishing the parent–child relationships. Subsequently, a new wave is initiated by the initializer robot. Upon receiving these messages, the remaining swarm members begin the motion that leads to either aggregation or dispersion.

The algorithm that regulates message transmission and reception during resizing can be represented by the state diagram shown in Fig. 2. As in the alignment task, each robot maintains two logical variables that guide the state transitions. The condition *res_p* indicates that the robot itself has reached a predefined resizing criterion, whereas the condition *res_f* indicates that its child robots have reached this criterion.

Three different types of messages are used during the resizing process, as summarized in Table 1. A message of type 7 is sent by a robot acting as a parent during the propagation wave that initiates the operation of its child robots. These children acknowledge reception by sending a message of the same type back to the parent. This exchange occurs in states $S_{AD}(01)$ and $S_{AD}(07)$ for a robot acting as a parent, and in states $S_{AD}(02)$ and $S_{AD}(08)$ when acting as a child. Message exchanges are repeated in states $S_{AD}(03)$, $S_{AD}(04)$, $S_{AD}(09)$, and $S_{AD}(10)$, but using messages of type 8. This procedure ensures that both parent and child robots confirm message reception. Messages of type 8 act as acknowledgments in a double-handshake scheme.

During the initial states, all robots execute the motion operation. This continues until a resizing condition is satisfied. This condition is defined by the parameter E, referred to as the *scaling factor*. Figure 3 illustrates the effect of E

on swarm resizing. Each robot is aware of the reference distance d_F relative to its child robots. This reference distance is measured during the initial recruitment subtask [27].

Table 1. Types of messages used during swarm resizing.

Message Type	msg_1	msg_2	msg_3	Message Type	msg_1	msg_2	msg_3
Propagation	id	–	7	Acknowledgment	id_{Parent}	id	8
Acknowledgment	id_{Parent}	id	7	Resizing confirmation	id	–	9
Propagation	id	–	8	Feedback	id_{Parent}	id	9

Robots evaluate their distance to their children in state $S_{AD}(04)$. Based on the measured values, they may transition to different states from $S_{AD}(05)$. If the resizing condition is satisfied for each child and the condition *res_p* is true, the robot stops moving and sends a type 9 message to its children. This message informs them that resizing has been achieved. Consequently, the robot's *res_f* condition becomes true, while the *res_p* condition of each child also becomes true. If resizing has not been achieved, two scenarios are possible. An intermediate robot for which both *res_p* and *res_f* are false returns to state $S_{AD}(07)$, operating again as a child in order to reposition itself at an appropriate distance from its parent. If *res_p* is true but *res_f* is false, the robot stops moving and returns to state $S_{AD}(01)$. Once the resizing conditions are satisfied, the initializer and intermediate robots transition from state $S_{AD}(05)$ to $S_{AD}(06)$, where they wait for a type 10 message. This message corresponds to a feedback signal sent by child robots, which are in state $S_{AD}(12)$. The algorithm terminates when all robots reach state $S_{AD}(13)$. At this point, the initializer can be certain that the swarm has successfully completed the resizing process.

4.3 Robot Movement

The confirmation that the swarm has reached the desired resizing condition depends on the distance between child robots and their respective parent, as illustrated in Fig. 3. However, if robots relied solely on this distance measurement, excessive proximity to other neighboring robots could occur. To address this issue, a motion rule was developed that takes into account the positions of all neighboring robots. In other words, each robot moves in response to the distances to all robots located within the sensing range of its detection system.

The proposed motion strategy is inspired by the flocking model introduced in Sect. 4.1. In this approach, the separation and cohesion rules are employed. The alignment rule was not required, as it is more suitable for situations in which the swarm as a whole must translate in space. Conversely, two additional rules were incorporated: one based on tracking the parent robot and another focused on obstacle avoidance.

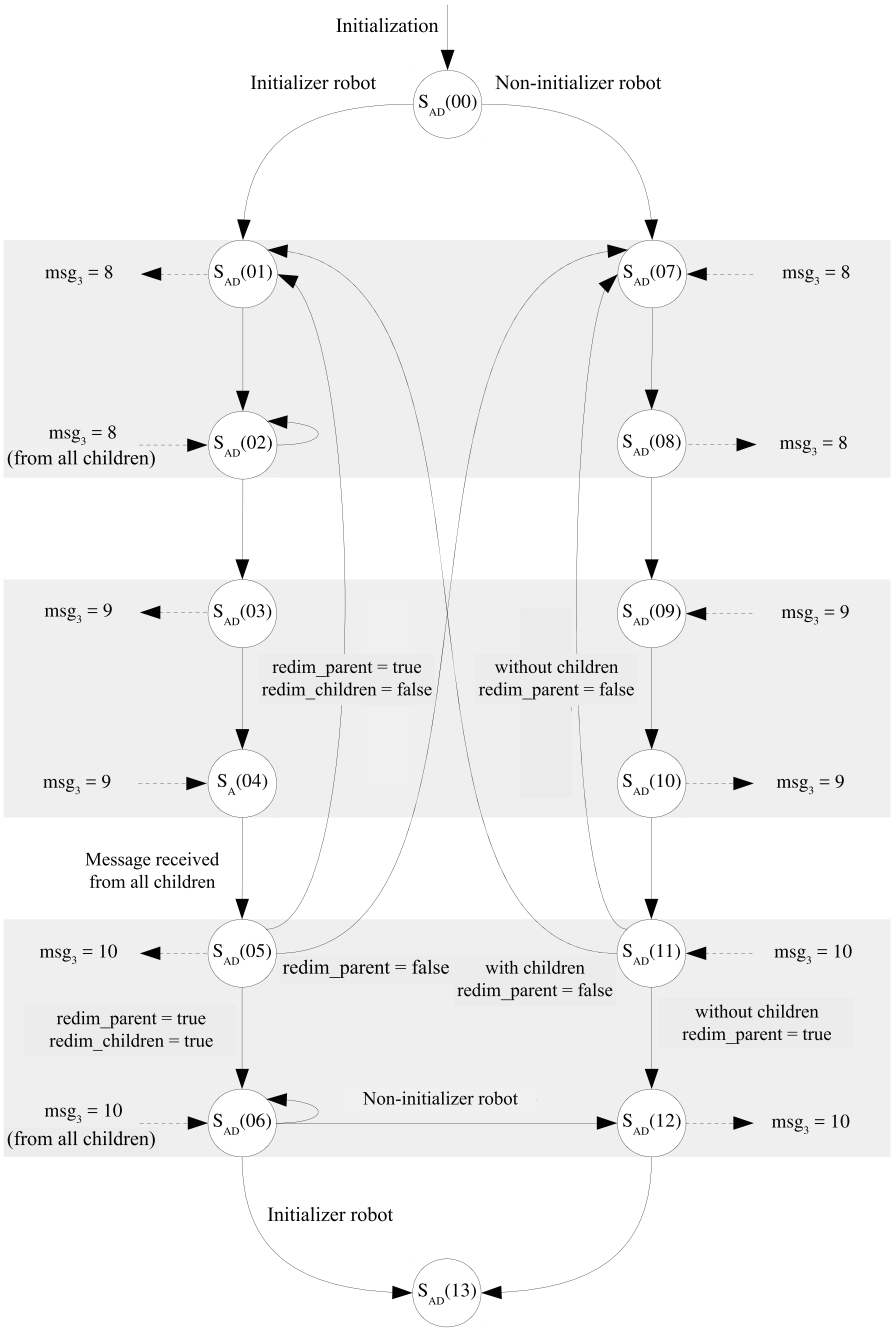
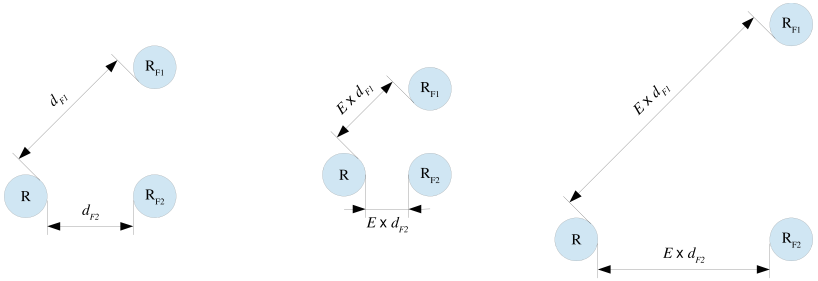


Fig. 2. State diagram describing the swarm resizing algorithm.



(a) Reference distances robot/children. (b) Swarm aggregation behavior with $E < 1$. (c) Swarm dispersion behavior with $E > 1$.

Fig. 3. Impact of the scaling factor E on swarm resizing.

For the cohesion behavior, each robot observes its neighborhood and computes the average position of the robots within its detection range, as described by Eq. 1. The cohesion distance $d_{x,y}(\text{coh})$ has (x, y) components defined in the coordinate frame of the robot performing the detection. The sum of the distances $d_{x,y}(\text{neighbor})$ is normalized by both the maximum sensing range (Sr) and the number of neighbors (Nn). As a result, each coordinate of the computed distance lies within the interval $[0, 1]$, where 0 corresponds to the minimum possible distance between robots and 1 represents the maximum detectable distance.

$$d_{x,y}(\text{coh}) = \frac{\sum_{i=1}^{Nn} d_{x,y}(\text{N}_i)}{Nn \times Sr}, \tag{1}$$

where Sr is the sensor radius and the N_i ith neighbor.

The distance $d_{x,y}(\text{coh})$ is used to compute the velocities of the robot's two motors. The expressions in Eq. 2 define four velocity components: two associated with the x-axis and two with the y-axis. It is important to note that the d_x component of the cohesion distance induces rotational motion, while the d_y component affects forward or backward motion. This behavior is analogous to that described the adjustment used for collective navigation [10].

$$\begin{aligned} velocity_{x1} &= v_{max} \times d_x & velocity_{y1} &= v_{max} \times d_y \\ velocity_{x2} &= -v_{max} \times d_x & velocity_{y2} &= v_{max} \times d_y \end{aligned} \tag{2}$$

The velocity of each motor is obtained by combining the velocity components associated with x and y, as expressed in Eq. 3. The velocity v_{cw} drives the motor responsible for clockwise rotation, whereas v_{ccw} controls the counterclockwise rotation, such that

$$\begin{aligned} v_{cw} &= 0.5 \times velocity_{x1} + 0.5 \times velocity_{y1} \\ v_{ccw} &= 0.5 \times velocity_{x2} + 0.5 \times velocity_{y2} \end{aligned} \tag{3}$$

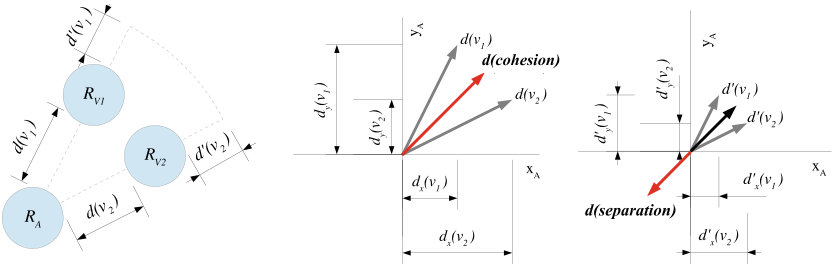
In the separation rule, the value used for velocity computation is not the inter-robot distance itself, but rather the distance $d'x, y(\text{neighbor})$ between a neighbor

and the maximum detection radius in that direction. This value is summed over all neighbors, resulting in the separation distance $dx, y(sep)$ defined in Eq. 4:

$$d_{x,y}(sep) = -\frac{\sum_{i=1}^{N_n} d'_{x,y}(\mathrm{N}_i)}{Nn \times Sr}, \tag{4}$$

where as before Nn is the number of neighbors, Sr is the sensor radius and N_i represents the ith neighbor.

Robots that are farther apart exert less repulsive force on one another. Conversely, robots closer to the detecting robot *i.e.*, farther from the detection boundary, generate a stronger repulsion, and thus the separation velocity should be maximal. The negative sign in Eq. 4 indicates that the resulting motion occurs in the opposite direction of the summed vector, as expected for a separation behavior. So, the same velocity equations used for cohesion can also be applied to separation. Both $d_{x,y}(coh)$ and $d_{x,y}(sep)$ are illustrated in Fig. 4.



(a) Robots R_{V1}/R_{V2} as neighbors of R_A (b) Distances between R_{V1}/R_{V2}—R_A. (c) Distances between R_{V1}/R_{V2}—R_A's vicinity

Fig. 4. Distances used in the computation of $d_{x,y}(coh)$ and $d_{x,y}(sep)$ for a robot R_A with two neighbors, R_{V1} and R_{V2}.

Two additional rules are employed to define robot velocities during swarm resizing. The first is associated with tracking the parent robot, preventing child robots from drifting too far away and potentially causing a loss of connectivity within the swarm. The second rule is obstacle avoidance, whose purpose is to prevent collisions when the mutual distance between robots becomes smaller than a predefined threshold. Both rules are similar to those used in the navigation task described in [10]. In this context, each rule results from the detection of a single neighbor: either the parent robot or a robot posing a collision risk. These two distance measurements, $d_{x,y}(\text{parent})$ and $d_{x,y}(collision)$, are converted into velocity components using Eq. 2.

The final velocity of each robot is obtained by combining the velocities generated by the four rules. In Eq. 5, the parameters w_1, w_2, w_3, and w_4 represent the weights associated with rules 1 through 4. The maximum possible values of v_{cw} and v_{ccw} are limited by v_{max}, which corresponds to the maximum speed

supported by the robot.

$$\begin{aligned} velocity_{cw} &= w_1 \times v_{cw}(1) + w_2 \times v_{cw}(2) + w_3 \times v_{cw}(3) + w_4 \times v_{cw}(4) \\ velocity_{ccw} &= w_1 \times v_{ccw}(1) + w_2 \times v_{ccw}(2) + w_3 \times v_{ccw}(3) + w_4 \times v_{ccw}(4) \end{aligned} \tag{5}$$

From Eq. 2, it can be observed that v_{max} is scaled by the distance computed in each rule. Since these distances are normalized and lie within the interval $[0, 1]$, the velocities v_{cw} and v_{ccw} associated with each rule are not maximal in most situations. As robots move closer to one another, $d_{x,y}(coh)$ decreases, thereby reducing the velocity contribution of the cohesion rule. A similar effect occurs for $d_{x,y}(sep)$ and the separation velocity when robots move farther apart.

5 Performance Results

The proposed rescaling task based on message waves is evaluated via simulations. The simulation time required for the aggregation and dispersion tasks are assessed, considering the impact of different factors, such as the number of robots and the inter-robot spacing, on the total execution time.

The swarms are composed of Kilobot [21] robots simulated in the CopealiaSim, also known earlier as V-REP [20]. The simulated *Kilobot* extends the physical robot's capabilities by increasing communication range, adding directional proximity sensing, and enabling more flexible motion. These enhancements are necessary to support message-wave algorithms and complex collective behaviors such as navigation, alignment, aggregation, and dispersion. All simulations reported in this work are carried out on computers equipped with an Intel Core i7-950 processor, operating at 3.0 GHz, 8 GB of memory, and running the Microsoft Windows 7 Home Premium operating system. Videos of the experiments can be found at https://l1nq.com/2gy6A.

5.1 Simulation Settings

The simulations are divided into two groups, corresponding to aggregation and dispersion experiments. In each experiment, the time required to complete the swarm rescaling ia reported. Depending on the task, this time is referred to as the Aggregation Time (τ_A) or the Dispersion Time (τ_D). Simulations are conducted with different numbers of robots, in order to analyze the relationship between execution time and swarm size. Let ρ_C be the critical path defined as the longest path from the initializer to a leaf robot in the swarm. During the simulations, the robots were disposed so that the number of robots in the critical path varies from 2 to 5.

The same local control algorithm is used for both aggregation and dispersion. The behaviors differ only in the flocking weights and in the rescaling factor E. For the aggregation, $0.5 < E < 1$ with a step of -0.1, meaning that the final parent-child distance is a fraction of the initial one, with a minimum of half the original distance. For dispersion, $1 < E < 2$ with a step of 0.2, causing robots

to increase their separation, with the maximum distance being twice the initial value.

The flocking weights are tuned differently for aggregation and dispersion, as they directly influence the contribution of each motion rule and, consequently, the swarm behavior. Table 2 summarizes the values used in the simulations. These parameters were selected so that the same set of values could be applied consistently across all aggregation and dispersion experiments.

All dispersion experiments were successfully performed using only the separation rule, which is the primary mechanism responsible for mutual repulsion among robots. Aggregation, however, required the combined use of all four motion rules with carefully chosen weights. In this case, the cohesion weight was set higher than the separation weight to promote convergence, while maintaining a nonzero separation component to avoid uneven motion caused by differences in robot velocities. This balance resulted in a more uniform aggregation process.

Table 2. Weights setup during the simulations

Rule	Obstacles	Cohesion	Separation	Parent
weight	w_1	w_2	w_3	w_4
Aggregation	1	3	0.5	3
Dispertion	0	0	3	0

These simulations allow the evaluation of execution time as a function of both swarm size ρ and the critical path length ρ_C. During the initial recruitment subtask, which establishes the parent-child relationships among robots, neighborhood detection is limited to a maximum of four neighbors. During the aggregation experiments, this restriction is not required, since robots are initially positioned far enough apart such that each robot can detect at most four neighbors. As the swarm aggregates, however, the number of robots within each local neighborhood naturally increases. During the dispersion experiments, robots start in close proximity, allowing neighborhoods to initially include more than four robots. As mutual repulsion occurs and the swarm spreads out, the neighborhood size decreases. By the end of the dispersion process, each robot detects at most four neighbors, which correspond to the same robots identified in its neighborhood at the beginning of the task.

Figure 5 shows the simulation times obtained for swarm aggregation. For experiments with the same number of robots, the bars represent the aggregation times obtained for different values of the rescaling factor E. Overall, a clear trend of approximately linear growth in τ_A with respect to swarm size can be observed. This behavior differs from that observed in the alignment simulations, where execution time is strongly dependent on the communication density ρ_C.

The observed difference is primarily due to robot motion. During alignment, movement consists solely of in-place rotations, whereas in aggregation the robots physically translate relative to one another. As the number of robots increases,

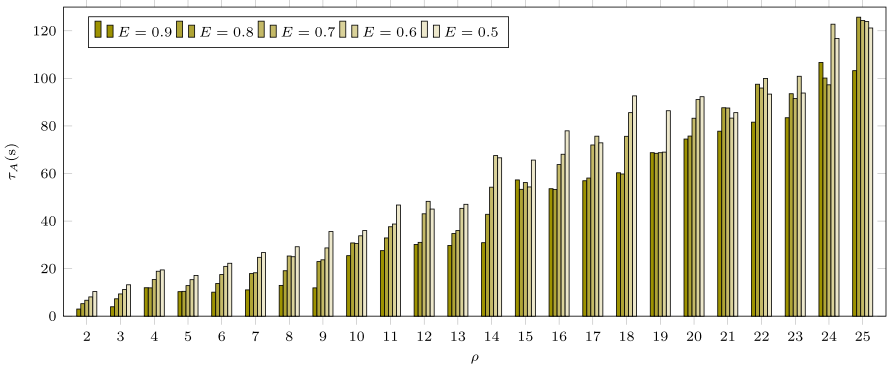


Fig. 5. Aggregation time for different swarm size and scaling factors.

more individuals participate in the flocking process, which directly affects each robot's velocity. Furthermore, as robots move closer together, the number of neighbors detected within each local neighborhood increases, further influencing motion dynamics. The results also highlight the variation of τ_A with respect to the rescaling factor E for experiments with the same swarm size. As expected, values of E closer to 1 generally lead to faster experiments, since smaller rescaling requires less robot movement. This trend is observed in most cases. However, for swarms with 12, 14, 15, 17, 21, 22, 23, 24, and 25 robots, some deviations from this behavior were observed. These exceptions can be attributed to non-uniform robot velocities during the aggregation process.

The results of the dispersion experiments are presented in Although the dispersion time τ_D increases with swarm size, this growth follows a pattern that differs from that observed in aggregation. In particular, τ_D consistently increases with the rescaling factor E for all swarm sizes considered. Moreover, this variation is more pronounced than the corresponding variation of τ_A observed in the aggregation experiments, as shown in Fig. 5.

The most notable behavior observed in the dispersion experiments is the oscillation of τ_D values. This effect is particularly evident in Fig. 6 for $E = 2$, especially in the cases with 10 and 21 robots. In the first case, τ_D increases more sharply when the swarm size changes from 9 to 10 robots than from 8 to 9. In the second case, τ_D decreases as the swarm size increases from 19 to 20 and from 20 to 21 robots. In both situations, this behavior can be attributed to the symmetry of the robot's disposition.

With 9 robots, the swarm configuration is symmetric, consisting of the initializer and 8 neighbors, where each of the four child robots has exactly one child of its own. With 10 robots, one of the initializer's children gains an additional child, breaking the symmetry. This asymmetry leads to an imbalance in the repulsive forces that govern dispersion, increasing the time required to reach the rescaling condition compared to the symmetric case. Conversely, when the swarm size increases from 20 to 21 robots, the added robot restores symme-

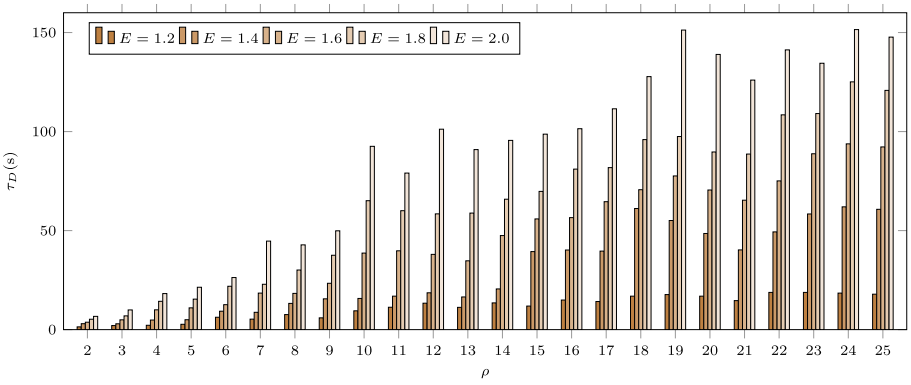
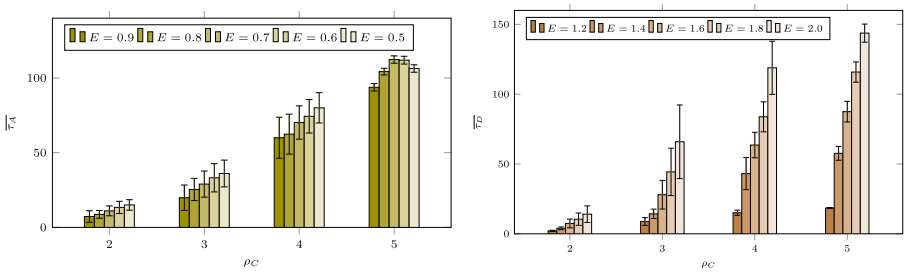


Fig. 6. Dispersion time for different swarm size and scaling factors.

try, which compensates for the increased swarm size and reduces the dispersion completion time.

Figure 7 shows Average aggregation and dispersion times, ($\overline{\tau_A}$) and ($\overline{\tau_D}$), respectively, considering the number of robots on the critical path, for different swarm size and scaling factors. These are computed as the mean simulation time for experiments with identical values of E and ρ_C.



(a) Average aggregation time (s) (b) Average dispersion time (s)

Fig. 7. Average aggregation and dispersion times considering the number of robots on the critical path, for different swarm size and scaling factors.

A significant variability in τ_A and τ_D alike is noticeable for $\rho_C = 3$ and $\rho_C = 4$, indicating a stronger sensitivity to swarm density under these conditions. The plot of $\overline{\tau_D}$ also highlights the influence of the rescaling factor E on dispersion time. For a given ρ_C, the values of $\overline{\tau_D}$ obtained with $E = 2$ are higher than those obtained with $E = 1.2$ for the immediately larger ρ_C. This behavior contrasts with the aggregation results, where $\overline{\tau_G}$ for a given ρ_C is always lower than all values corresponding to the next larger ρ_C, regardless of E.

6 Conclusions

This work employs a message-wave algorithm as the core mechanism for designing collective swarm behaviors. The main requirement for this approach is the presence of a message-based communication system, in which robots can exchange messages with neighbors within a defined communication range. Neighborhood relationships must be bidirectional: if a robot can send messages to a neighbor, it must also be able to receive messages from that same neighbor. These local communication links define the swarm's communication topology.

The focus of this study is swarm resizing, encompassing aggregation and dispersion behaviors. These tasks require an initial recruitment phase to establish parent-child relationships among robots. Robot motion is governed by flocking rules, which regulate attraction and repulsion relative to neighbors. While message waves coordinate task initiation and termination through the communication system, flocking-based motion relies on the detection system. The interaction between these two systems enables precise control over the degree of swarm resizing.

A natural extension of this work is a deeper investigation of the implemented resizing task. For instance, one may analyze the impact of additional factors on execution time, such as the number of neighboring robots. In the current experiments, robots were arranged in a grid topology with at most four neighbors per robot, although the task was shown to function under other spatial configurations. Exploring alternative regular topologies, such as those with three or six neighbors, is therefore of interest.

Acknowledgment. This work is supported by Fundação Carlos Chagas Filho de Amparo à Pesquisa do Estado do Rio de Janeiro (FAPERJ – Brazil) with grant numbers 201.013/2022 and 200.434/2026. This work is also supported by Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq – Brazil). We are most grateful for their continuous financial support.

References

1. Bayındır, L.: A review of swarm robotics tasks. Neurocomputing **172**, 292–321 (2016). https://doi.org/10.1016/j.neucom.2015.05.116
2. Bonabeau, E., Theraulaz, G., Deneubourg, J.L., Aron, S., Camazine, S.: Self-organization in social insects. Trends Ecol. Evol. **12**(5), 188–193 (1997)
3. Brambilla, M., Ferrante, E., Birattari, M., Dorigo, M.: Swarm robotics: a review from the swarm engineering perspective. Swarm Intell. **7**(1), 1–41 (2013)
4. Broecker, B., Caliskanelli, I., Tuyls, K., Sklar, E., Hennes, D.: Social insect-inspired multi-robot coverage. In: Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems, pp. 1775–1776. International Foundation for Autonomous Agents and Multiagent Systems (2015)
5. Cao, Y.U., Fukunaga, A., Kahng, A.: Cooperative mobile robotics: antecedents and directions. Auton. Robot. **4**(1), 7–27 (1997)

6. Faigl, J., Kulich, M.: On benchmarking of frontier-based multi-robot exploration strategies. In: Proceedings of the 2015 European Conference on Mobile Robots, pp. 1–8. IEEE (2015)
7. Flocchini, P., Prencipe, G., Santoro, N., Widmayer, P.: Arbitrary pattern formation by asynchronous, anonymous, oblivious robots. Theoret. Comput. Sci. **407**(1), 412–447 (2008)
8. Garnier, S., Gautrais, J., Asadpour, M., Jost, C., Theraulaz, G.: Self-organized aggregation triggers collective decision making in a group of cockroach-like robots. Adapt. Behav. **17**(2), 109–133 (2009)
9. Iocchi, L., Nardi, D., Salerno, M.: Reactivity and deliberation: a survey on multi-robot systems, pp. 9–32. Springer (2001)
10. Junior, L.S., Nedjah, N.: Efficient strategy for collective navigation control in swarm robotics. Procedia Comput. Sci. **80**, 814–823 (2016). https://doi.org/10.1016/j.procs.2016.05.371
11. Liu, W., Winfield, A.: Self-assembly in heterogeneous modular robots, pp. 219–232. Springer (2014)
12. Mendonça, R., Nedjah, N., Mourelle, L.: Efficient distributed algorithm of dynamic task assignment for swarm robotics. Neurocomputing **172**, 345–355 (2016)
13. Nedjah, N., Cardoso, K.D.S., Mourelle, L.D.M.: Collaborative swarm robotics for object transport via caging. Sensors **25**(16) (2025). https://doi.org/10.3390/s25165063. https://www.mdpi.com/1424-8220/25/16/5063
14. Nedjah, N., Ferreira, G.B., de Macedo Mourelle, L.: Swarm robotics for collaborative object transport using a pushing strategy. Expert Syst. Appl. **271**, 126610 (2025). https://doi.org/10.1016/j.eswa.2025.126610
15. Nedjah, N., Junior, L.S.: Review of methodologies and tasks in swarm robotics towards standardization. Swarm Evol. Comput. **50**, 100565 (2019). https://doi.org/10.1016/j.swevo.2019.100565
16. Nedjah, N., Mendonça, R., Mourelle, L.: PSO-based distributed algorithm for dynamic task allocation in a robotic swarm. Procedia Comput. Sci. **51**, 326–335 (2015)
17. Nedjah, N., Ribeiro, L.M., de Macedo Mourelle, L.: Communication optimization for efficient dynamic task allocation in swarm robotics. Appl. Soft Comput. **105**, 107297 (2021). https://doi.org/10.1016/j.asoc.2021.107297
18. Reynolds, C.: Flocks, herds and schools: a distributed behavioral model. ACM SIGGRAPH Comput. Graph. **21**, 25–34 (1987)
19. Ribeiro, L.M., Nedjah, N., de Carvalho, P.V.R.: Optimized navigation for slam using marker-assisted region scanning, path finding, and mapping completion control. Appl. Sci. **15**(7) (2025). https://doi.org/10.3390/app15073433. https://www.mdpi.com/2076-3417/15/7/3433
20. Rohmer, E., Singh, S., Freese, M.: V-rep: a versatile and scalable robot simulation framework. In: Proceedings of the 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 1321–1326. IEEE (2013)
21. Rubenstein, M., Ahler, C., Hoff, N., Cabrera, A., Nagpal, R.: Kilobot: a low cost robot with scalable operations designed for collective behaviors. Robot. Auton. Syst. **62**(7), 966–975 (2014)
22. Sá, A., Nedjah, N., Mourelle, L.: Distributed efficient localization in swarm robotic systems using swarm intelligence algorithms. Neurocomputing **172**, 322–336 (2016)
23. Sá, A., Nedjah, N., Mourelle, L.: Distributed efficient localization in swarm robotics using min-max and particle swarm optimization. Expert Syst. Appl. **50**, 55–65 (2016)

24. Schmickl, T., Hamann, H.: Beeclust: a swarm algorithm derived from honeybees. Bio-inspired Comput. Commun. Netw. (2011)
25. Segall, A.: Distributed network protocols. IEEE Trans. Inf. Theory **29**(1), 23–35 (1983)
26. Silva, L., Nedjah, N.: Wave algorithm for recruitment in swarm robotics. In: Gervasi, O., et al. (eds.) Computational Science and Its Applications - ICCSA 2015, pp. 3–13. Springer, Cham (2015)
27. Silva Junior, L., Nedjah, N.: Distributed strategy for robots recruitment in swarm-based systems. Int. J. Bio-inspired Comput. **8**, 99–108 (2016). https://doi.org/10.1504/IJBIC.2016.076336
28. Silva Junior, L., Nedjah, N.: Wave algorithm applied to collective navigation of robotic swarms. Appl. Soft Comput. **57**, 698–707 (2017). https://doi.org/10.1016/j.asoc.2016.06.004
29. Winfield, A.: Foraging robots, pp. 3682–3700. Springer (2009)