International Journal of Intelligent Robotics and Applications (2025) 9:804–814
https://doi.org/10.1007/s41315-025-00442-6

REGULAR PAPER

Robot swarm aggregation using an improved Beeclust method

Y. Yuliana Rios[1] · Oscar Acevedo[2] · Luis Luis García[3]

Received: 16 January 2024 / Accepted: 28 July 2024 / Published online: 3 May 2025

Abstract
Swarm robotics is a topic within multi-robot systems that aims to solve engineering problems by drawing inspiration from the behavior observed in nature by social animals such as ants, bees, fish school, among others. The main challenge in these systems is the design of the controller, which must operate at the level of the individual robot to achieve results at the level of the entire swarm. In previous works, various basic behaviors of social insects have been studied and classified, which have been adapted to the field of robotics to emulate and use in the implementation of controllers and problem-solving. Examples of these behaviors include aggregation, dispersion, and resource searching (foraging). In this project, a simulation implemented in Matlab is presented, which implements a modified version of the Beeclust algorithm, which emulates the aggregation behavior of newly born bees. The modifications made focus on two basic actions of the algorithm: linear velocity and rotation angle. These adjustments have resulted in a 30% improvement in the average aggregation time compared to the original algorithm.

Keywords Robotics · Beeclust · Aggregation · Robot swarm · Optimization

1 Introduction

Swarm robotics endeavors to create intelligent systems comprised of multiple entities that collaborate to achieve a common objective (Akopov et al. 2020; Al-Obaidy and Al-Azawi 2019). This concept draws inspiration from nature's observations, including organisms like ants, birds, fish, among others. Through their collective efforts, these organisms demonstrate the ability to accomplish results that would be challenging to attain individually.

To explain how a swarm accomplishes a desired objective, researchers have developed fundamental concepts such as self-organization and collective intelligence. These concepts provide an understanding of how group-level behaviors (macroscopic) emerge from local interactions between the entities and their immediate environment (microscopic).

Furthermore, key properties have been identified in these swarms, including local interaction among entities and their immediate environment, local communication, and decentralized control. These characteristics confer two crucial attributes to swarms: robustness and scalability. Robustness refers to the swarm's ability to achieve its objective even in the presence of entity loss. Scalability, on the other hand, pertains to the swarm's capacity to maintain or enhance its efficiency as the number of swarm members increases. However, in practice, various factors impose limitations on these characteristics and, consequently, the maximum achievable efficiency (Sahin 2005).

Building upon these studies, swarm robotics has been effectively applied in both simulated environments and physical implementations, utilizing the developed models. These endeavors have led to the identification of several fundamental behaviors exhibited by swarms, including aggregation, dispersion, pattern formation, flocking, and object clustering (Bayindir 2016).

Oscar Acevedo and Luis Luis García contributed equally to this work.

✉ Y. Yuliana Rios
yeriosdi@uis.edu.co

Oscar Acevedo
oacevedo@utb.edu.co

Luis Luis García
luis.garciag@unisimon.edu.co

[1] Escuela de Ingeniería Mecánica, Universidad Industrial de Santander, Bucaramanga 100190, Santander, Colombia

[2] Facultad de Ingenierías, Universidad Tecnológica de Bolívar, Cartagena 10587, Bolívar, Colombia

[3] Ingeniería Biomédica, Universidad Simón Bolívar-Audacia, Barranquilla 610101, Atlántico, Colombia

It is worth emphasizing that certain fundamental behaviors within a swarm require the preceding execution of other behaviors to enhance their efficiency. For example, in the case of monitoring, it is essential for robots to cluster at a specific location within the area to be explored to ensure adequate coverage. In the context of a robot swarm, this clustering process is known as aggregation. Aggregation can be driven by environmental cues or determined autonomously by the swarm, referred to as self-organization (Hamann 2018).

One of the challenges that arises when physically implementing these behaviors is the development of the robot control system. This system is devised at an individual level, within each robot, with the goal of attaining collective behavior at the swarm level. In broad terms, the controller must define both individual and social actions for the robot. Individual actions encompass factors such as movement type, sensory perception, and relevant events specific to the robot. Conversely, social actions pertain to how the robot should interact with other robots and the conditions under which it should distance itself from them. The controller must make informed decisions to facilitate cooperation and coordination among swarm members, thus enabling the achievement of shared objectives.

Some controller implementations follow models based on probabilistic finite state machines, which clearly define actions and the probabilities of transitioning or continuing with those actions. Other implementations incorporate concepts of artificial physics to define the overall behavior of the robot. Additionally, controllers generated from neural networks have been proposed, commonly optimized through evolutionary methods. Methodologies, such as the one presented in Duarte et al. (2016), have also been proposed, which suggest dividing the overall swarm behavior into a combination of basic behaviors, thereby generating two-level controllers. In this approach, the first level is responsible for individually implementing the basic behaviors, while the second level combines these behaviors through a higher-level methodology.

In this article, we examine the behavior of an algorithm called Beeclust (Kernbach et al. 2009) aimed at achieving environment-driven aggregation. Experiments are performed in different scenarios, and modifications of the original algorithm are presented to improve efficiency.

2 Review

This section summarizes previous work on swarm robotics, focusing on the Beeclust algorithm for promoting environment-driven robot aggregation.

The early works related to swarm robotics date back to the 1990s. However, it was around 2005 when a significant surge in research on robot swarms was observed. One notable example of such work is the study published by Sahin in 2005, which identified the key properties of robot swarms (Sahin 2005). In 2009, Kernbach presented one of the most prominent algorithms for implementing environment-guided robot aggregation, known as Beeclust (Kernbach et al. 2009). This algorithm draws inspiration from the collaborative behavior of newly hatched bees, collectively searching for the warmest point in the beehive. Beeclust stands out for its robustness and simplicity, enabling swift implementation on physical robots. The results presented by the authors, as well as by other researchers investigating this algorithm, demonstrate its effectiveness in both simulations and real robot implementations. Consequently, Beeclust has become a benchmark for studies in the same field. Noteworthy among these works is the 2009 study by Schmickl, which focused on analyzing the algorithm's behavior in response to environmental changes (Schmickl et al. 2009).

Several studies have been dedicated to improving the efficiency of the Beeclust algorithm. These enhancements can be divided into two main groups: those focusing on improving the internal actions of the algorithm, and those seeking to enhance the capabilities of the robots, the environment, and the algorithm itself to improve decision-making.

The first group of improvements aims to reduce the randomness component of algorithms, which can be a significant problem, by integrating hardware and software that enable more accurate decision-making. An example of this is the work of Arvin et al. In Arvin et al. (2012), where the robot's speed, waiting time, and rotation angle were changed to improve the aggregation probability. This study and another by Arvin et al. (2014) used a series of sensors and fuzzy logic to determine the optimal rotation angle of the robot.

In the second group of improvements, the focus is on enhancing the capabilities of the robots, the environment, and the algorithm to improve decision-making. For example, the work presented by Arvin et al. (2018) incorporates the use of pheromones to "attract" other robots towards a specific group, thereby increasing the probability of aggregation. Another example involves the placement of markers in the environment to assist the robots in locating the hottest point and clustering around it, as described in the study by Amjadi et al. (2021). These additional enhancements in the environment and the robots' capabilities contribute to improving the efficiency and effectiveness of the Beeclust algorithm, but increase its cost.

Another area of research is devoted to modeling the behavior of the entire robot swarm, often utilizing models that depict flow behavior. Notable works in this field include the compartment model proposed by Hamann et al. (2008), which captures the swarm's spatial interactions at

a macro-level, and the Stock-and-Flow model discussed by Schmickl et al. (2009), which offers an alternative representation of collective behavior.

Furthermore, robot simulators have been utilized to study and model the behavior of swarms that employ Beeclust as their primary controller. For example, the work by Bodi et al. (2012) delves into the interaction between robots and the environment using the SMARS simulator. Similarly, Sakira et al. present a bio-inspired approach utilizing the PLAYER/STAGE robot simulator in their study (Ramroop et al. 2018).

All these modeling and simulation approaches offer a more profound understanding of the collective behavior exhibited by robot swarms, enabling exploration of various scenarios and conditions to assess the effectiveness of the Beeclust algorithm as the primary controller. The following section introduces the Beeclust algorithm and details its implementation.

3 Beeclust

The Beeclust algorithm, introduced by Kernbach et al. (2009), is inspired by the behavior of newly born bees. These bees naturally seek out the warmest point in the hive for survival. Despite their inherent limitations, they collaborate to find the hot spot. The results presented in Kernbach et al. (2009) demonstrate that the proposed algorithm accurately captures the behavior of bees. It can be applied to robot swarms, enabling them to exhibit the same behavior and achieve aggregation at a specific point in the environment with a desired property, such as the highest temperature. The Beeclust algorithm is outlined in Algorithm 1.

It is important to highlight that in the Beeclust algorithm, the rotation angle is determined randomly, which simplifies its implementation but can result in behaviors deviating from the optimal rotation angle towards the hottest point. During the conducted experiments, instances have been observed where the robot selects rotation angles that lead it away from the hottest point, even when it is in close proximity to it. This indicates that there are situations in which the algorithm may generate sub-optimal outcomes in terms of directing towards the point of highest temperature.

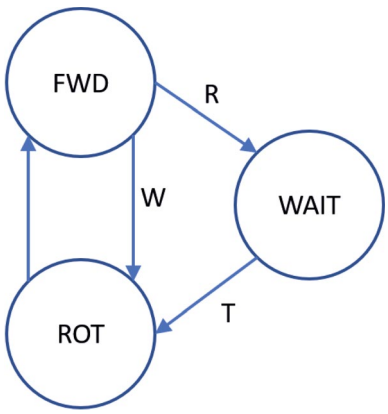
The behavior of the Beeclust algorithm can be categorized into two distinct types: individual behavior and social behavior. Individual behavior involves a "random walk" movement pattern with the ability to navigate around obstacles. It is worth noting that, in this work, the term "random walk" refers to the primary objective of moving forward in a straight line and only altering the direction upon encountering an obstacle. During this behavior, the robot lacks awareness of its environment and focuses primarily on seeking other robots. On the other hand, social behavior is triggered when the robot detects the presence of another robot. In such instances, the robot modifies its behavior and temporarily halts its movement based on the temperature of the encountered point. A specific function is employed to calculate the duration of this pause (Kernbach et al. 2009):

$$time = t_{max} \cdot \frac{s^2(t)}{s^2(t) + \theta} \tag{1}$$

where t_{max} is the maximum waiting time, $s(t)$ is the source sensor value and θ defines the slope of the curve. The parameter values were obtained from preliminary experiments.

Algorithm 1 Beeclust algorithm

```
1:  Move forward
2:  if obstacle detected then
3:      stop motion
4:      if obstacle is wall then
5:          go to step 15
6:      end if
7:      if obstacle is robot then
8:          go to step 12
9:      end if
10: end if
11: go to 1
12: Read sensor
13: Compute delay
14: Wait until timeout
15: Rotate randomly
16: Go to step 1
```



The time the robot stops is directly linked to the temperature of the collision point. The higher the temperature, the longer the robots will remain stationary. This, in turn, increases the likelihood of other robots finding the stopped ones, facilitating the formation of groups and enhancing the probability of encounters. This phenomenon exemplifies a positive feedback loop, which is one of the concepts that contribute to explaining the aggregation process of a robot swarm.

After the waiting time is completed, the robot concludes its social behavior and seeks to distance itself from the group through random rotation. Although this behavior may seem contradictory to aggregation, it actually enables the swarm to maintain a dynamic behavior, explore the environment, and prioritize optimal points instead of getting stuck in local maxima. During the conducted experiments, it was observed that robots within the group can become "trapped" by other robots, leading to a new waiting period and prolonging the lifespan of the group. On the other hand, robots on the periphery of the group frequently modify their behavior, joining and then separating from the group. This continuous cycle of aggregation and dispersion contributes to the dynamics and adaptability of the swarm in response to changes in the environment.

It is worth emphasizing that the Beeclust algorithm not only achieves robot aggregation but also exhibits desirable characteristics for robot swarms. First, explicit communication between robots is not required. This means that each robot makes decisions based solely on its own environment and local interactions (detection) with other robots, enabling a simpler and more efficient implementation. Second, Beeclust is highly scalable, allowing it to adapt to swarms of different sizes without requiring additional global adjustments or parameters. Third, it does not depend on global information or prior knowledge of the environment, making it suitable for dynamic or unknown environments. And finally, it has decentralized control. The algorithm focuses on the individual actions of each robot in response to local events, without the need for a centralized supervisor. This allows for greater autonomy and flexibility within the swarm, as each robot can independently respond to its immediate environment.

4 Proposed upgrades

As mentioned earlier, several efforts have been made to enhance the performance of the Beeclust algorithm. These enhancements can be classified into two main categories: internal and external. Internal improvements focus on optimizing the robot's inherent actions, while external enhancements introduce additional capabilities to enhance the robot's perception or add new skills.

Internal improvements aim to fine-tune the parameters of the algorithm's three core actions: linear displacement speed, waiting time, and rotation angle. In Arvin et al. (2012), the linear speed is modulated based on the ambient temperature, with higher temperatures resulting in reduced speed. Similarly, the waiting time is adjusted in relation to the number of detected neighboring robots, allowing for a longer waiting time when more neighbors are present. In Arvin et al. (2012) and Hamann et al. (2012), the random rotation angle is replaced by an estimated value derived from a fuzzy logic system.

Examples of external improvements encompass the integration of additional hardware that augments the robots' capabilities in terms of aggregation. For instance, in the work cited as Arvin et al. (2018), "pheromones" are employed to enhance the aggregation process. By emitting these signals, robots can attract and guide others towards specific groups, thereby increasing the likelihood of aggregation. Similarly, in the study (Amjadi et al. 2021), the implementation of beacons serves as reference points for the robots' orientation, enabling them to locate aggregation zones more efficiently.

The analysis of these studies reveals an improvement in aggregation time compared to the original algorithm. However, implementing these improvements introduces additional costs in terms of hardware and energy consumption required for the robot to perform the additional functionalities. While it is plausible that certain robot implementations may already possess the necessary hardware, it is generally advisable to minimize resources and simplify the structure of swarm robot deployments to enable scalability. Nonetheless, this guideline is not absolute, and the feasibility of incorporating these enhancements should be assessed based on the specific requirements and constraints of the application.

In this work, we propose two internal adaptations that aim to reduce the aggregation time of the swarm without the need for additional hardware beyond what is already used in the original Beeclust algorithm. These adaptations also aim to reduce the energy overhead associated with the improvements made in other studies. Specifically, we introduce adjustments to the velocity and rotation angle of the robots, leveraging existing hardware capabilities while utilizing memory to store relevant robot measurements. The following sections describe each improvement in detail.

4.1 Angle of rotation

The proposed improvement is based on estimating the direction of the point of maximum temperature using already available sensor information, rather than selecting a random turning direction. This enhancement builds upon the

general operation of the original Beeclust algorithm. During the robot's movement, it travels in a straight line from the last collision point until it encounters a new collision. By storing the measured temperature at the last collision point and comparing it with the temperature at the current collision point, it is possible to determine an estimated direction of the maximum temperature. Figure 1 presents the idea. Initially, the blue robot was located at position B, alongside the other robots. However, after its waiting time ended, the robot turned in the direction indicated by the green arrow, separating from the group, and colliding with the wall at point A. In order for the robot to reunite with the others, it must turn towards the direction indicated by the yellow arrow. This estimation can be achieved if the robot knows the temperature at the previous collision point (B) and the current point (A), subsequently calculating a gradient based on these measurements.

The new function for estimating the turning angle will be located between lines 3 and 4 on Algorithm 1. Algorithm 2 showcases the implemented function.

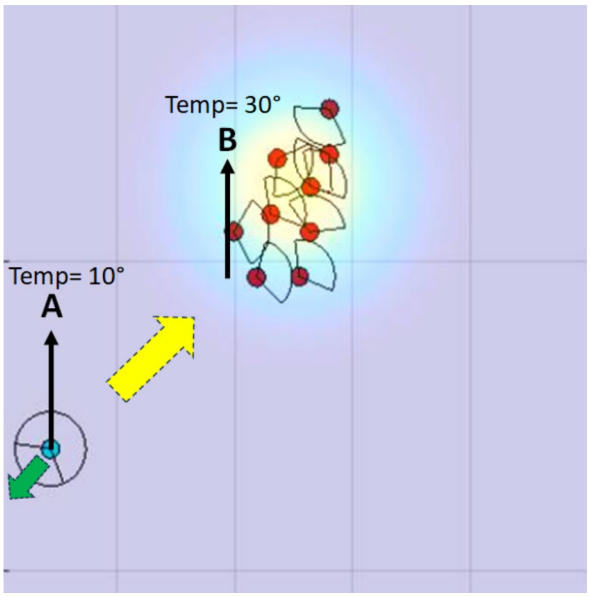


Fig. 1 Basic idea for angle of rotation

Algorithm 2 Rotate forward or backward function

1: $delta = T_{actual} - T_{old}$
2: **if** $abs(delta) < \delta$ **then**
3: Rotate forward angle + rand[-180°, +180°]
4: **else**
5: **if** delta > 0 **then**
6: Keep forward angle + rand[-90°, +90°]
7: **else**
8: Rotate 180° + rand[-30°, +30°]
9: **end if**
10: **end if**

Function 2 begins by calculating the difference between the current temperature and the last recorded temperature. The sign of this difference provides an estimate of the general direction the robot should take: maintain the current direction or switch to the opposite direction. As depicted in the algorithm (lines 5–8), a random component is added to each direction to provide flexibility in the robot's behavior. It's worth noting that the experiments carried out only involve a single point of maximum temperature, and most of the arena maintains a similar temperature. This leads to a small temperature difference, which could trigger undesired reactions in the robot. Consequently, a threshold was established for the temperature difference (δ in line 2) to guide the robot's directional decision. If this difference doesn't exceed the threshold, the robot opts for a random angle of turn, like the original Beeclust algorithm. To conclude, the parameter values used in this algorithm were determined through preliminary experiments.

Table 1 The robot's velocity based on the measured temperature

Temperature Range	Speed
Low (< 60%)	High (0.4 m/s)
Middle (60–75%)	Middle (0.2 m/s)
High (> 75%)	Slow (0.1 m/s)

Table 2 Simulation parameters

Parameter	Value
Arena size	1.0 m × 1.5 m
Hotspot	$(0.5X_{max}, 0.75Y_{max})$
Swarm size	12 robots
Simulation success	11 robots

4.2 Speed

In this study, we propose a modification to the methodology presented in Arvin et al. (2012), where the robot continuously measures the ambient temperature and adjusts its speed accordingly. In our work, temperature measurement and speed adjustment are performed exclusively during collision events. This approach aims to reduce the aggregation time compared to the original Beeclust algorithm while minimizing the energy cost, as observed in the previous proposal (Arvin et al. 2012).

Our modification is based on the understanding that the temperature remains relatively constant throughout the arena, except in the zone where the maximum temperature point is located. Consequently, continuous temperature measurements prove to be largely unnecessary and energy-consuming. However, when a robot comes to a stop in the high-temperature zone following a collision with another robot, we desire the robot to move slowly within this area. By doing so, we extend its duration in the high-temperature zone, thereby increasing the probability of collisions with other robots. The modified version introduced in this study facilitates the realization of this objective. The proposed upgrade of Beeclust algorithm modifies line 15 in Algorithm 1.

To determine the robot's velocity based on the measured temperature, we rely on Table 1 as a reference. This table is constructed using as guideline the information presented in Arvin et al. (2012).

To enhance the algorithm's adaptability to various target physical variables, we have expressed the information in Table 1 in terms of values generated by the AD converter and their corresponding range. As shown in Table 1, we have divided the AD converter value range into three zones and assigned a specific velocity value to each zone. The boundaries of these zones and the assigned velocities were determined through preliminary experiments.

During the preliminary experiments, it was also observed that the robot was capable of exiting the high-temperature zone at a low speed and traversing a significant distance in the arena before colliding with another object. To address

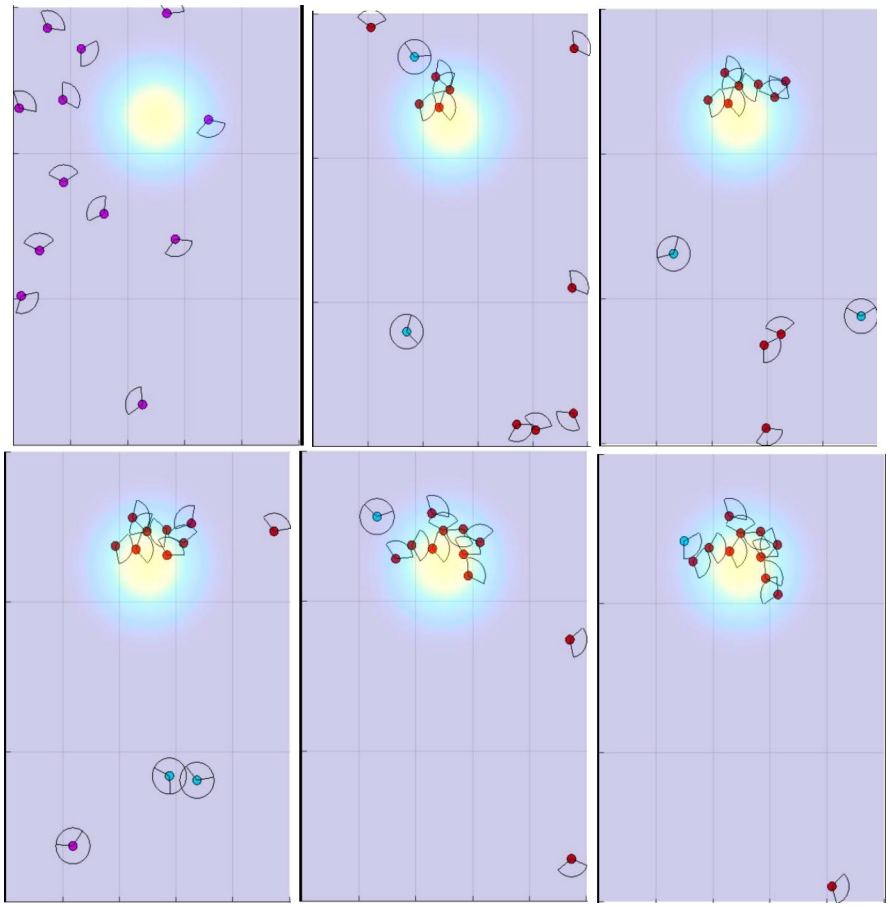


Fig. 2 Simulation screenshot: begin top left, end bottom rigth

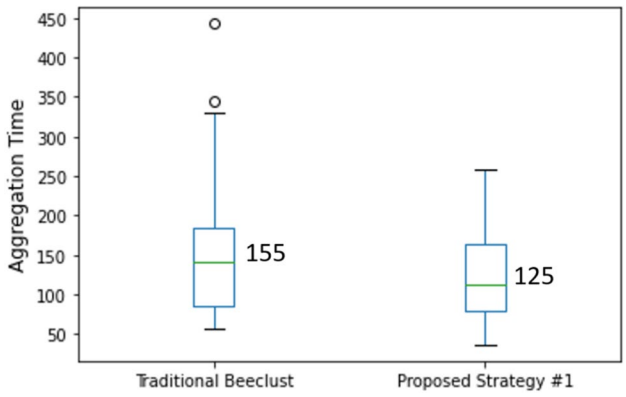


Fig. 3 Comparison of rotation angle and original Beeclust algorithm

this behavior, a timer was implemented to reset the velocity to a predetermined value if no collision occurred within a certain time period.

4.3 Speed and rotation angle

The ideas presented in the previous sections can be combined with minimal adjustments to the original Beeclust algorithm, thus resulting in the final version of the proposed enhancements in this work. Algorithm 3 illustrates the location of the implemented modifications.

Table 3 Statistics for rotation angle

Algorithm	Average (s)	Variance	Coef. var (%)
Beeclust	155.02	7802.06	56.98
Proposed	125.8	3311.4	45.75

5 Results

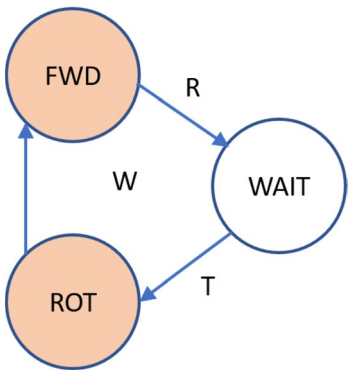
The implementation of the proposed enhancements can be seamlessly integrated into the existing robot used for the original Beeclust algorithm, as they do not require any additional hardware beyond the utilization of memory for storing the latest temperature measurement and a timer for resetting the robot's velocity to its default value. This timer can be readily implemented in software, ensuring a straightforward and efficient implementation process.

In this study, a modified version of the simulator introduced in the articles Acevedo et al. (2022a, 2022b) was utilized. In this adapted version, adjustments were made to the robot model to incorporate the desired functionality, and the collection of additional measurements was enabled to facilitate statistical comparisons.

Since a physical implementation was not carried out, and detailed model information from (Arvin et al. 2012) is not available, we compared the results obtained using our methodology with the original Beeclust algorithm. However, we developed a dedicated model in our simulator specifically to evaluate the speed improvement presented in the work (Arvin et al. 2012). Therefore, the results for this specific enhancement are also included in our analysis. Please note that the developed model may not fully represent the

Algorithm 3 Beeclust algorithm: change speed according to temperature

```
1: Move forward
2: if obstacle detected then
3:     stop motion
4:     Update speed
5:     if obstacle is wall then
6:         go to step 16
7:     end if
8:     if obstacle is robot then
9:         go to step 13
10:    end if
11: end if
12: go to 1
13: Read sensor
14: Compute delay
15: Wait until timeout
16: Rotate forward or backward
17: Go to step 1
```



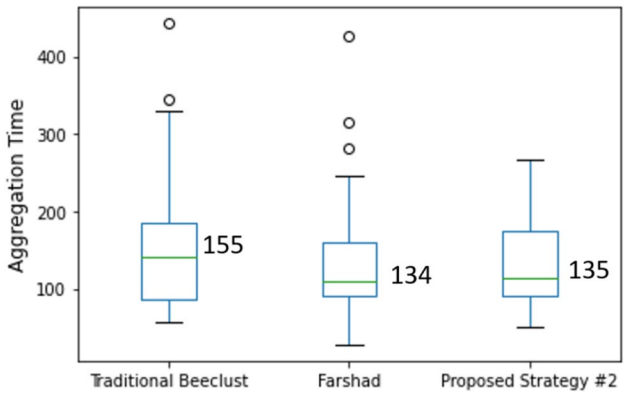


Fig. 4 Comparison of speed adjust with Farshad (Arvin et al. 2012) and the original Beeclust algorithm

Table 4 Statistics for speed adjustment

Algorithm	Average (s)	Variance	Coef. var (%)
Beeclust	155.02	7802.06	56.98
Farshad (Arvin et al. 2012)	135.45	6305.61	58.63
Proposed	134.42	3279.43	42.60

methodology of Arvin et al. (2012). We made an effort to model it as closely as possible within the constraints of the simulator used and the available information.

The experiments presented are focused on quantifying the swarm's aggregation time near the hottest point within the arena. Considering the dynamic characteristics of the Beeclust algorithm, success is defined as robots reaching the target, where a minimum of 90% of the robots should be located within the designated area. Table 2 presents additional simulation parameters and Fig. 2 presents a simulation screenshot.

5.1 Rotation angle experiment

Figure 3 presents the results obtained for the improvement in the rotation angle compared to the results obtained from the original Beeclust algorithm.

As shown in Fig. 3, the original Beeclust algorithm exhibits considerable variability in its results due to its strong random component. On the contrary, our algorithm achieves a lower average, as well as reduced variability. Table 3 presents statistical results.

Although the results reflect an improvement compared to the original Beeclust algorithm, there is also noticeable variability. This variability can be understood by examining Fig. 2, which illustrates how the majority of the arena has a low and similar temperature. Consequently, a robot moving through these zones will choose to make random turns, just like Beeclust does. However, once the robot manages to get closer to the high-temperature zone, it will be able to estimate the direction towards this zone as it moves away, accelerating its aggregation process. Therefore, it is to be expected that both algorithms exhibit similar behavior in the initial phase of the simulation. However, over time, our algorithm tends to achieve faster aggregation due to its ability to estimate the direction towards the area of interest.

Table 5 Statistics for speed adjustment

Algorithm	% Increase
Farshad (Arvin et al. 2012)	7998
Proposed	98.7

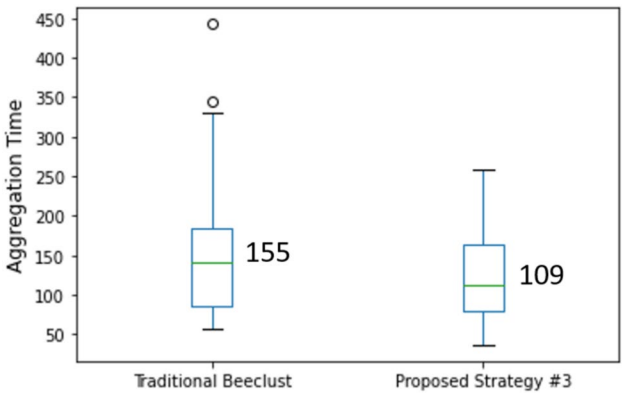


Fig. 5 Comparison of rotation angle and speed adjust and original Beeclust algorithm

Table 6 Statistics for speed and rotation angle adjustment

Algorithm	Average (s)	Variance	Coef. var (%)
Beeclust	155.02	7802.06	56.98
Proposed	109.42	2568.26	46.32

Regarding the increase in energy cost due to the additional use of the temperature sensor, the simulator results reveal an average increase of 141% in the utilization of this sensor by the improved algorithm compared to the original Beeclust algorithm. This excess consumption arises from the additional measurements that the improved algorithm performs before executing a rotation, a procedure not carried out in the original version of the algorithm.

Given the current state of simulator implementation, it is challenging to precisely quantify the impact of this additional energy cost on the robots. Nevertheless, leveraging the insights offered by the simulator, it becomes feasible to make suitable adjustments to the robot's energy system before deploying the swarm, aiming to mitigate any potential adverse effects.

Table 7 ANOVA results for the data

F	P-value	F crit
8.023	0.005875	3.96

5.2 Speed experiment

In this experiment, two metrics are observed: aggregation time and energy cost associated with the use of the temperature sensor. The comparison is made among the original Beeclust algorithm, our implementation of the algorithm proposed in Arvin et al. (2012), and the version proposed in this work. Figure 4 presents the results for these methodologies.

As can be observed, the results somehow show a similar pattern, with the original algorithm displaying higher variability compared to the proposed enhancements, which exhibit more consistent outcomes. The statistical summary of the results is presented in Table 4.

Based on the information provided in Table 4, it is evident that the proposed enhancements show a lower average aggregation time, with our work yielding results very similar to those of Arvin et al. (2012).

It is worth noting that these enhancements solely focus on changing speed. The decision regarding the turning angle remains the same as in the original algorithm, which is random, leading to a significant variability in the obtained results.

Also, it is important to consider that the proposed enhancements also incur an additional energy cost due to the continuous utilization of the temperature sensor for speed adjustment. Table 5 provides an overview of the number of temperature measurements for each algorithm.

As observed in Table 5, the energy overhead of methodology (Arvin et al. 2012) is considerably higher when compared to our work. As an illustrative example, a robot enhanced by Arvin et al. (2012) utilized the temperature sensor 4223 times, whereas the Beeclust algorithm with the same behavior employed it only 37 times. This is expected since algorithm (Arvin et al. 2012) has a high temperature sampling frequency (50 samples/s. in our simulation). On the contrary, the algorithm presented in this study demonstrates substantially reduced sensor usage (twice as low as Beeclust) while still maintaining a similar average aggregation time compared to Arvin et al. (2012). However, it is important to note that these results are based on our interpretation and implementation of the work in Arvin et al. (2012), therefore, it is not advisable to generalize these findings yet.

5.3 Rotation angle and speed experiment

The final experiment involved combining the two previous enhancements and comparing the performance against the original Beeclust algorithm. This combination was seamlessly implemented as there was no interference between the actions of the two improvements. The results obtained are presented in Fig. 5.

As shown in Fig. 5, our algorithm presents a significant reduction about 30% in the average aggregation time. Also, the variability is lower when compared to the original algorithm. Table 6 displays mean and variance values.

Performing an ANOVA test on the data with alpha 0.05 yields the results shown in Table 7.

Table 7 indicates that the combined algorithm effectively delivers a substantial improvement in swarm aggregation times when compared to the original Beeclust algorithm, within the context of the employed simulator's working environment.

Regarding the excess energy consumption derived from the use of the temperature sensor, the relationship remains consistent with that obtained in the experiment detailed in the previous experiments, showing an 86.6% increase in usage compared to the Beeclust algorithm.

6 Conclusion

This study introduces two enhancements to the original Beeclust algorithm. The first improvement involves adjusting the linear movement speed, which remains constant in the original version of the algorithm. In this work, the speed is discretely adjusted based on the detected temperature, decreasing as the temperature increases. The second enhancement focuses on adjusting the rotation angle. While in the original algorithm this angle is random, in this work, three possibilities are considered: forward movement, backward movement, or random rotation. This choice is based on comparing the current temperature to the last recorded measurement. As a result of these enhancements, an average 30% reduction in the time required for aggregation is achieved compared to the original algorithm.

The implemented improvements do not require additional hardware compared to that used by robots employing the original algorithm. In terms of software, there is a slight increase in memory usage to store an additional data point, though this increase is negligible. There is an uptick in energy consumption, as the data shows an average doubling in the usage of the temperature sensor. Despite not having been quantified, it is reasonable to assume that the increase in the robot's total energy cost will not be significantly affected. This is because temperature sensors used in these types of robots typically have a moderate energy cost, and furthermore, the reduction in the time needed to complete the task must be

considered, resulting in an overall lower energy expenditure for the robot.

In terms of future research, the implementation of the proposed algorithm on actual robots is on its way, as well as the exploration of the use of more sophisticated algorithms to optimize the efficiency of the original algorithm. The plan includes constructing approximately 20 robots and a dedicated work environment tailored for implementing the algorithms outlined in this paper. Within this setting, light will serve as the attracting factor instead of temperature, as the latter presents additional complexities.

Acknowledgements The authors thank the support of Universidad Industrial de Santander, Colombia, through Project 3770 (Projects supported by Fondo de la Vicerrecoría de Investigación y Extensión).

Funding Not applicable.

Availability of data and materials Not applicable.

Declarations

Ethical approval Not applicable.

References

Acevedo, O., Rios, Y.Y., Duque, J., Gomez, E., García, L.: A software for simulating robot swarm aggregation. In: Figueroa-García, J.C., Franco, C., Díaz-Gutierrez, Y., Hernández-Pérez, G. (eds.) Applied Computer Sciences in Engineering, pp. 386–399. Springer, Cham (2022a)

Acevedo, O., Rios, Y.Y., García, L., Narvaez, D.: A study of the Beeclust algorithm for robot swarm aggregation. In: 2022 IEEE International Conference on Machine Learning and Applied Network Technologies (ICMLANT), pp. 1–6 (2022b). https://doi.org/10.1109/ICMLANT56191.2022.9996514

Akopov, A.S., Beklaryan, L.A., Beklaryan, A.L.: Cluster-based optimization of an evacuation process using a parallel bi-objective real-coded genetic algorithm. Cybern. Inf. Technol. **20**(3), 45–63 (2020)

Al-Obaidy, M., Al-Azawi, R.: Cluster-based algorithm for energy optimization of swarmed robots using swarm intelligence. In: 2019 Sixth HCT Information Technology Trends (ITT), pp. 202–207. IEEE (2019)

Amjadi, A.S., Raoufi, M., Turgut, A.E.: A self-adaptive landmark-based aggregation method for robot swarms. Adapt. Behav. (2021). https://doi.org/10.1177/1059712320985543

Arvin, F., Samsudin, K., Ramli, A.R., Bekravi, M.: Imitation of honeybee aggregation with collective behavior of swarm robots. Int. J. Comput. Intell. Syst. **4**, 739–748 (2012). https://doi.org/10.1080/18756891.2011.9727825

Arvin, F., Turgut, A.E., Bazyari, F., Arikan, K.B., Bellotto, N., Yue, S.: Cue-based aggregation with a mobile robot swarm: a novel fuzzy-based method. Adapt. Behav. **22**, 189–206 (2014). https://doi.org/10.1177/1059712314528009

Arvin, F., Turgut, A.E., Krajnik, T., Rahimi, S., Okay, I.E., Yue, S., Watson, S., Lennox, B.: Phi Clust: pheromone-based aggregation for robotic swarms. In: IEEE International Conference on Intelligent Robots and Systems, pp. 4288–4294 (2018). https://doi.org/10.1109/IROS.2018.8593961

Bayindir, L.: A review of swarm robotics tasks. Neurocomputing **172**, 292–321 (2016). https://doi.org/10.1016/j.neucom.2015.05.116

Bodi, M., Thenius, R., Szopek, M., Schmickl, T., Crailsheim, K.: Interaction of robot swarms using the honeybee-inspired control algorithm BEECLUST. Math. Comput. Model. Dyn. Syst. **18**, 87–100 (2012). https://doi.org/10.1080/13873954.2011.601420

Duarte, M., Costa, V., Gomes, J., Rodrigues, T., Silva, F., Oliveira, S.M., Christensen, A.L.: Evolution of collective behaviors for a real swarm of aquatic surface robots. PLoS One **11**(3), 0151834 (2016)

Hamann, H.: Swarm robotics: a formal approach. Swarm Robotics: A Formal Approach (2018). https://doi.org/10.1007/978-3-319-74528-2

Hamann, H., Schmickl, T., Wörn, H., Crailsheim, K.: Analysis of emergent symmetry breaking in collective decision making. Neural Comput. Appl. **21**, 207–218 (2012)

Hamann, H., Wörn, H., Crailsheim, K., Schmickl, T.: Spatial macroscopic models of a bio-inspired robotic swarm algorithm. In: 2008 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS, pp. 1415–1420 (2008). https://doi.org/10.1109/IROS.2008.4651038

Kernbach, S., Thenius, R., Kernbach, O., Schmickl, T.: Re-embodiment of honeybee aggregation behavior in an artificial micro-robotic system. Adapt. Behav. **17**, 237–259 (2009). https://doi.org/10.1177/1059712309104966

Ramroop, S., Arvin, F., Watson, S., Carrasco-Gomez, J., Lennox, B.: A bio-inspired aggregation with robot swarm using real and simulated mobile robots. In: Giuliani, M., Assaf, T., Giannaccini, M.E. (eds.) Towards Autonomous Robotic Systems, pp. 317–329. Springer, Cham (2018)

Şahin, E.: Swarm robotics: from sources of inspiration to domains of application. In: Swarm Robotics, pp. 10–20. Springer, Berlin, Heidelberg (2005)

Schmickl, T., Hamann, H., Worn, H., Crailsheim, K.: Two different approaches to a macroscopic model of a bio-inspired robotic swarm. Robot. Autonom. Syst. **57**, 913–921 (2009a). https://doi.org/10.1016/J.ROBOT.2009.06.002

Schmickl, T., Thenius, R., Moeslinger, C., Radspieler, G., Kernbach, S., Szymanski, M., Crailsheim, K.: Get in touch: cooperative decision making based on robot-to-robot collisions. Autonom. Agents Multi-Agent Syst. **18**(1), 133–155 (2009b)

Y. Yuliana Rios received Ph.D. degrees in electrical engineering from the Advanced Studies and Research Center of the National Polytechnic Institute (CINVESTAV-IPN), Guadalajara Campus, Mexico, in 2019. She is currently an associate professor of mechanical engineering at the Universidad Industrial de Santander, Bucaramanga, Colombia. Her research interests include intelligent control, biomedical systems, inverse optimal control, and robotics swarms.

Oscar Acevedo holds a Ph.D. in Electrical and Computer Engineering from Southern Illinois University, USA. Currently, he serves as an Associate Professor at Universidad Tecnológica de Bolívar in Colombia, where he teaches courses in electronics and embedded systems. His research interests include wireless sensor networks and robotic swarms.

Luis Luis García holds a Master's degree in Engineering from the estadual de santa catarina, Brasil. He currently works as a phd student in Engineering program at universidad politécnica de madrid, in spain. His areas of interest include control, dynamic, and robotics.