How to Evolve Aggregation in Robotic Multi-Agent Systems

Paolo Pagliuca[1,*,†], Giuseppe Trivisano[2,†] and Alessandra Vitanza[1,†]

[1]Institute of Cognitive Sciences and Technologies, National Research Council (CNR-ISTC), Rome, Italy
[2]Università degli Studi di Bari Aldo Moro, Bari, Italy

Abstract

Multi-Agent Systems are characterized by multiple agents interacting to solve tasks that may be difficult, or even impossible, for a single agent. While discovering solutions to problems with a single objective might be relatively straightforward, the picture changes when coping with Multi-Objective Optimization (MOO), where problems require the simultaneous optimization of multiple objectives that potentially conflict with each other. This is particularly relevant in Multi-Agent Systems (MASs), since each agent's behavior affects the overall system performance. For example, the capability of a system, composed of many robots, to both locomote and aggregate simultaneously requires the definition of appropriate fitness measures and the usage of suitable algorithms. In this work, we investigate the conditions necessary to promote aggregation in a robotic MAS, with a particular focus on how conflicting objectives can hinder the learning of effective behaviors. Specifically, we designed a novel fitness function and tested it in a relatively simple aggregation scenario. Furthermore, we considered a recently introduced MOO problem, in which a MAS of five robots must develop the ability to aggregate while in motion. Our outcomes show that, despite the challenges in designing effective fitness functions, the proposed formulation successfully supports aggregation in the simpler scenario and enhances aggregation capabilities in the more complex one.

Keywords

Multi-Agent Systems, Multi-Objective Optimization, Evolutionary Algorithms, OpenAI-ES, Aggregation

1. Introduction

Multi-Agent Systems (MASs) [1–3] are characterized by the presence of multiple intelligent entities, termed as "agents", that coexist in a shared environment and can interact, thus reciprocally influencing. As a result of these local interactions, MASs are capable of developing very complex behaviors and can solve problems that are not feasible for a single agent. In particular, a MAS requires the presence of at least two agents (i.e., a two-agent system or dyad). Thanks to the interaction between agents, the whole system can give rise to sophisticated behaviors. Examples of complex strategies discovered in robotic MASs include foraging [4–11], aggregation [12–18] and predator avoidance [19–23]. A subfield of MASs is swarm robotics [24–26], where a group of agents (e.g., robots or drones) performs tasks not feasible for a single element. The idea is to mimic biological systems like colonies of ants, herds of sheep, schools of fish or flocks of birds [27, 28].

Generally, the development of the agents' skills in a MAS is often achieved through Evolutionary Algorithms (EAs) [29], i.e., optimization methods that are inspired from biological evolution and are capable of providing solutions to a broad set of problems, such as classic control [30–32], robot navigation [33–35], foraging [4, 36], function optimization [37, 38], or competitive co-evolution [39, 40].

EAs have proven to be valuable tools to cope with Multi-Objective Optimization (MOO) problems [41–44], in which multiple conflicting objectives must be optimized simultaneously [45]. In such scenarios, findings solutions that satisfy all objectives is very difficult. Therefore, Pareto optimality

WOA 2025: 26th Workshop "From Objects to Agents", July 2–5, 2025, Trento, Italy
*Corresponding author.
†The authors contributed equally.
paolo.pagliuca@istc.cnr.it (P. Pagliuca); g.trivisano@alumni.uniba.it (G. Trivisano); alessandra.vitanza@istc.cnr.it (A. Vitanza)
0000-0002-3780-3347 (P. Pagliuca); 0000-0002-7760-8167 (A. Vitanza)

[46] is used to identify a set of valuable solutions from which the experimenter can select the best. MOO in a MAS is a significantly complex scenario, where multiple agents must interact in order to optimize different, often conflicting, goals. Finding solutions to this kind of problems is far from trivial. As illustrated in [47], the authors employed OpenAI-ES (OpenAI Evolutionary Strategy) [48] to evolve two behaviors, locomotion and aggregation, in a swarm of five Ant Pybullet robots [49]. Although the evolved agents exhibited good locomotion capability, they failed to develop any form of aggregation. This underscores the difficulty of simultaneously optimizing multiple objectives, especially in MASs. Moreover, the study emphasizes that the definition of the fitness function is paramount in these kinds of problems.

Building on these insights and taking inspiration from this previous study, this work explores how these conflicting objectives interact to shape the evolution of distinct behaviors in a MAS. To this end, we first introduced a new fitness function specifically tailored to promote aggregation and tested it in a simple robot aggregation scenario. Next, we adapt the fitness function originally proposed in [47] by replacing only its aggregation reward component with our new formulation. Our results show that ***(a)*** in the simple robotic aggregation scenario, the new fitness function efficiently evolves effective aggregation behaviors; ***(b)*** in the AntBullet Swarm scenario, the new adapted function shows improvements in aggregation, although the swarm does not yet display fully coordinated aggregation.

The rest of the paper starts with an overview of related works in the field of MAS, EAs and MOO (Section 2), with a specific focus on aggregation. Then, the problems addressed and the experimental settings are described (Section 3). In Section 4, we present the quantitative and qualitative outcomes of our experiments. Finally, Section 5 contains our final remarks and possible future research directions.

2. Background

Aggregation is a process in which individuals gather in groups for specific purposes [50], such as protection against predators [51] or speeding up for food [52]. This phenomenon is frequently observed in nature, both in microorganisms (e.g., *Dictyostelium discoideum* [53], *Capsaspora owczarzaki* [54] and *Brachionus calyciflorus* [55]) and in complex animals (e.g., flocks of birds [56] and schools of fish [57, 58]). Aggregation represents one of the fundamental collective behaviors, as it gives rise to various cooperative behaviors [18], such as coordinated movement [59], self-assembly [60] or collective transport [61]. A specific body of research focuses on self-organized aggregation [18], in which the individuals of the group aggregate autonomously, without any central control. This kind of aggregation allows the development of control systems that are robust to partial failure, flexible and scalable, and can be performed using only local interactions between individuals. In biological systems, self-organized aggregation manifests through either positive or negative feedback mechanisms [18]: the former can occur as an attraction force towards a given signal source, while the latter acts as a regulatory or repulsive mechanism between individuals.

The effectiveness and evolutionary importance of this behavior have prompted research to replicate it in simulated environments, following the principles of swarm robotics [25, 26]. Several studies have used evolutionary algorithms [29], and specifically evolutionary strategies [62, 63], to develop aggregation behaviors in MASs or robot swarms. Evolutionary algorithms — translating the principles of natural evolution into algorithms — are promising solutions for the automatic learning of complex control policies that are difficult to design manually from scratch. In [64], different algorithms, including CMA-ES (Covariance Matrix Adaptation Evolution Strategy) [65], GA (Genetic Algorithm) [66] and OpenAI-ES [48], were compared on an aggregation task. The study evaluates convergence time, policy quality, scalability and generalization for swarms of different sizes. In general, all the algorithms proved to be effective in completing the task, with differences in the stability of the aggregated cluster and the aggregation times, especially for smaller swarms (5, 10 and 20 robots). In [15], CMA-ES and xNES (Exponential Natural Evolution Strategies) [67] were compared on a specific aggregation task, highlighting the importance of communication. Similarly, a comparison of the different aggregation behaviors evolved by CMA-ES, xNES and OpenAI-ES under different experimental conditions is proposed

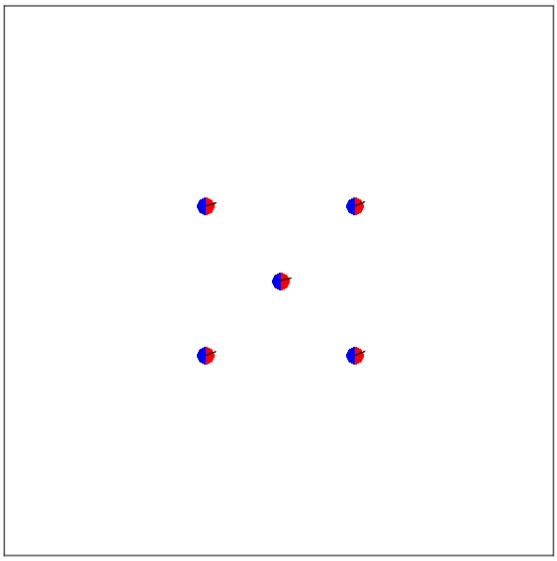


Figure 1: The environment in the robot aggregation problem. Robots are initially located in specific positions, forming a cross. The index of the central robot is 1. Starting from the robot placed in the upper right corner, the indices are progressively increased in a counter-clockwise direction.

in [16]. Finally, the paper [68] demonstrates the validity of an automatic approach based on evolutionary optimization via PSO (Particle Swarm Optimization) [69] to design interpretable and scalable PFSM (Probabilistic Finite State Machines) controllers for the fundamental task of aggregation in swarm robotics. The use of this type of controllers offers greater policy interpretability and potentially better transferability to real-world robotic environments compared to the typically used neural network controllers.

The works discussed so far focus on the optimization of a single objective, known as Mono-Objective Optimization. In such problems, there exists at least one optimal solution, as well as multiple equivalent solutions. Instead, when the problem involves multiple and potentially conflicting objectives, as previously mentioned, we speak of Multi-Objective Optimization (MOO) [45]. Compared to the previous scenario, solving a MOO problem poses significant challenges, since the aim is to obtain the optimal solution for all the objectives simultaneously, which is not feasible. In this case, the possible solutions, called Pareto-optimal [46], are potentially infinite and lie on the Pareto front. The choice of one of these solutions depends on the subjective preferences of a human decision maker. In contrast to the problem presented in [47], another approach that investigated the development of aggregation, in conjunction with other tasks, is the one illustrated in [70], where a decentralized algorithm for robotic swarms based on limited local interactions is introduced and applied to a MOO problem in which agents are rewarded for their ability to aggregate and avoid obstacles. The approach, tested both in simulation and in real-world settings, has been proven to be effective in developing cohesive behavior and dynamic obstacle avoidance.

3. Materials and Methods

3.1. Robot Aggregation problem

The first scenario involves a group of 5 MarXbot robots [71] that are placed in a square-walled arena of $5m \times 5m$. The initial configuration of the robots is shown in Fig. 1: the central robot is located at the center of a theoretical square, whose vertices represent the initial positions of the other robots. The radius of each MarXbot robot (R_R) is $8.5cm$. The position of the central robot, labeled with index 1, is $(p_1^x, p_1^y) = (2.5m, 2.5m)$. The initial locations of the other robots, whose indices are increased

counter-clockwise, are determined according to the following rules (Eq. 1):

$$\begin{aligned} p_2^x &= p_1^x + \Delta x \qquad p_2^y = p_1^y + \Delta y \\ p_3^x &= p_1^x - \Delta x \qquad p_3^y = p_1^y + \Delta y \\ p_4^x &= p_1^x - \Delta x \qquad p_4^y = p_1^y - \Delta y \\ p_5^x &= p_1^x + \Delta x \qquad p_5^y = p_1^y - \Delta y \end{aligned} \tag{1}$$

where Δx and Δy are set to $R_R \times 8 = 68$cm. As a consequence, the distance between central and peripheral robots is approximately 0.962 meters, and the distance between each pair of peripheral robots measures 1.36 meters.

The robots are equipped with 8 infrared sectors for wall detection and an omni-directional camera consisting of 4 sectors of 90° each. The camera enables robots to detect colored objects in the environment. Additionally, each robot can turn on/off both frontal and rear LEDs (red and blue, respectively) in order to signal its presence to the other robots.

The robots' goal is to aggregate within the arena. To this end, we defined the fitness function as reported in Eqs. 2 - 4:

$$\hat{d}_i = \frac{1}{N_R - 1} \sum_{j=1}^{N_R - 1} | \, d_T - d_{i,j} \, | \tag{2}$$

$$D_i = M \times e^{-\frac{\hat{d}_i^2}{\sigma^2}} \tag{3}$$

$$F_a = \frac{1}{N_R} \sum_{i=1}^{N_R} D_i \tag{4}$$

In Eqs. 2 - 4, the symbol $d_{i,j}$ indicates the distance between the agents i and j (with $i \neq j$), $\hat{d}_i$ is the resulting average distance for the $i-th$ agent and N_R is the number of robots. The d_T parameter represents the target distance considered sufficient to solve the problem (in our experiments, we set $d_T = 1.5mm$). The symbols M and σ denote, respectively, the maximum value achieved by the function F_a when the target distance is reached and the standard deviation of the Gaussian function. As regards the experiments reported here, we used the values $M = 7$ and $\sigma = 8$. Finally, the symbol D_i represents the distance reward (see Fig. 2). This function was chosen empirically after a preliminary investigation phase (see Fig. 3), during which we observed that D_i is characterized by a more gradual slope than alternative functions, while still maintaining the desired maximization.

A feed-forward neural network controls each robot. The network has 12 inputs, one layer of 10 hidden neurons and 4 outputs. The hidden and output neurons have biases, and their activation function is the *tanh*. The infrared sensors and the camera provide input data feeding the neural network, which performs its computation and produces two outputs to control the wheel speeds, and two outputs to control the frontal and rear LEDs.

3.2. AntBullet Swarm problem

The second scenario is notably more challenging and requires dealing with a MOO problem. In particular, we consider the problem introduced in [47], which involves a MAS composed of 5 AntBullet robots that must aggregate while locomoting in an unbounded environment. A snapshot of the problem is provided in Fig. 4.

The agents' goal is to learn how to aggregate and locomote. Due to the pitfalls highlighted in [47], we applied the following modifications:

- we split the evolutionary process into two phases and adopted an incremental evolution approach [72]. During the first phase, agents learn only to locomote; in the second phase, instead, they have to aggregate while locomoting;

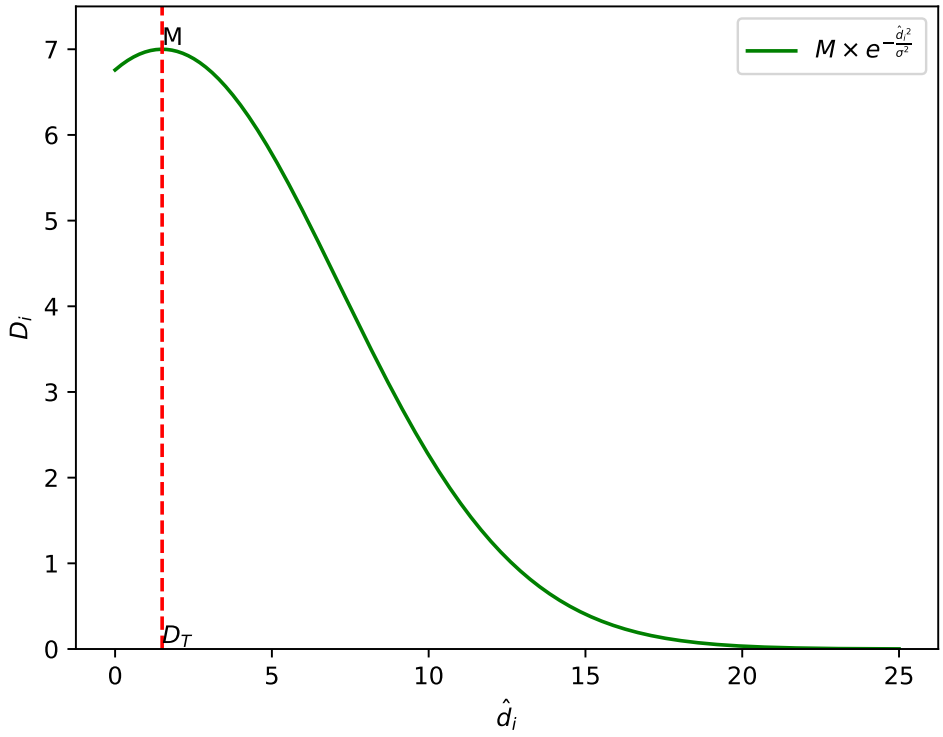


Figure 2: Distance reward D_i is a function of the average distance of robot i from the other robots. The curve is plotted with $M = 7$, $\sigma = 8$ and $d_T = 1.5$.

- we modified the aggregation reward function by using the one defined in Eq. 3;
- during the second phase, the reward for locomotion is notably reduced — but not removed — in order to make the agents exploit the skills acquired in the first phase.

This latter decision was made because existing studies [34, 73] have shown that training agents to learn two different skills sequentially may lead to a "vanishing" phenomenon, where previously acquired capabilities are forgotten.

Therefore, with the aim to promote the evolution of successful strategies, the fitness function has been designed according to Eqs. 5 - 7:

$$F_{ant} = \begin{cases} F_1 & \text{if } s < \frac{N_S}{2} \\ \\ F_2 & \text{if } s \geq \frac{N_S}{2} \end{cases} \tag{5}$$

$$F_1 = \frac{1}{N_R} \sum_{i=1}^{N_R} P_i + S_i + J_i \tag{6}$$

$$F_2 = \frac{1}{N_R} \sum_{i=1}^{N_R} 0.1 \times P_i + D_i + S_i + J_i \tag{7}$$

The symbol N_S in Eq. 5 represents the total number of evaluation steps performed during evolution. The symbols P_i, S_i and J_i in Eqs. 6 - 7 indicate, respectively, the progress reward (i.e., how much the agent i locomotes), the stall cost related to the magnitude of the actions performed by agent i, and the number of joints extended at their limits. In Eq. 7, D_i is the distance reward computed according to Eq. 3 (differently from the previous scenario, here we set the target distance d_T to 1.5 meters, coherently with [47]). We modified this component to improve the performance of the aggregation shown in [47], aiming to address the challenges of MOO more effectively. In fact, as pointed out by the authors,

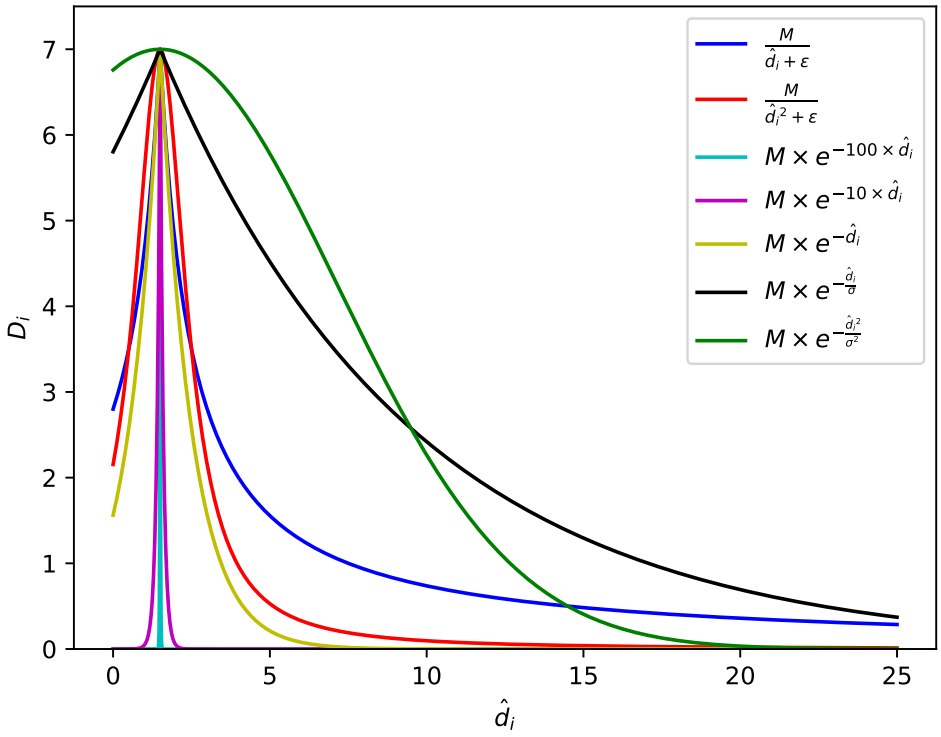


Figure 3: List of distance reward functions empirically tested in order to choose D_i. The curves are plotted with $M = 7$, $\sigma = 8$ and $\varepsilon = 1$.

the original fitness function (see [47], Eqs. 5-6) was ineffective due to the different magnitudes of the components P_i and D_i. Moreover, locomotion is easier to achieve than aggregation: a robot can locomote regardless of the others, whereas aggregation involves the need to interact and coordinate with others. Consequently, the outcomes presented in [47] show that agents, evolved with OpenAI-ES, are characterized by good locomotion capabilities, but do not aggregate.

In addition to the fitness modification, we endowed the robots with an omni-directional camera capable of detecting other robots within an 8-meter range. The camera view is split into 6 sectors of 60° each. This sensor has been designed to make each robot capable of perceiving the presence of nearby mates, hence encouraging the evolution of more effective aggregation behaviors. The information provided by the camera input for a generic robot r is defined as follows: if robot r detects a teammate m, the corresponding sector s is activated and returns a value defined according to Eq. 8:

$$cam_s = (1.0 - \frac{d_{r,m}}{d_{MAX}}) \tag{8}$$

where the symbol $d_{r,m}$ indicates the distance between the robots r and m and d_{MAX} is the maximum distance (with $d_{MAX} = 8$). A detailed list of the robot's equipment is provided in Table 1.

The robot controller is a feed-forward neural network with 34 input neurons, an internal layer containing 20 hidden neurons, and 8 output neurons. All neurons have associated biases. The activation function of hidden neurons is the *tanh* function, while the activation function of output neurons is the *linear* function.

3.3. Experimental setting

For both scenarios, the OpenAI Evolutionary Strategy (OpenAI-ES) [48] was employed, as it represents a modern and quite sophisticated algorithm for evolving successful locomotion [48, 74, 75] and aggregation [14, 16, 64] behaviors. OpenAI-ES works by initializing a single solution, called *centroid*, which encodes the connection weights of the neural network controller determining the robot's behaviors. The centroid

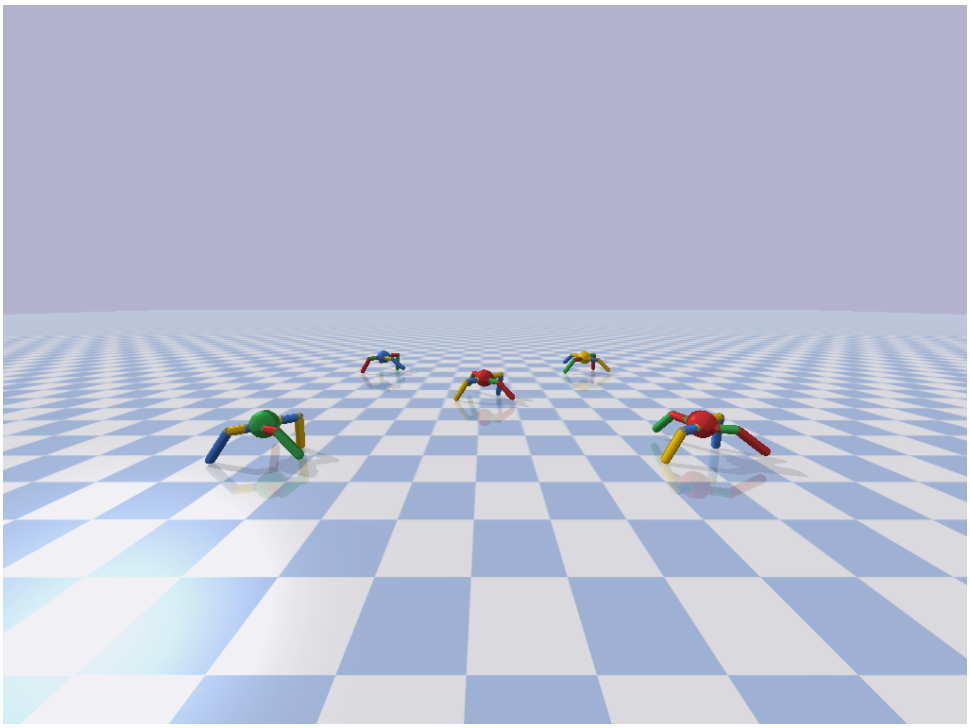


Figure 4: The AntBullet Swarm problem. The group consists of 5 AntBullet robots, initially in a cross formation. The environment is unbounded, allowing the agents to move freely.

Table 1
Robot's equipment (i.e., sensors and motors). The symbol z indicates the agent's height, while the symbol z_{init} represents the initial height of the agent. The symbol $angle_{to\_target}$ denotes the relative angle between the robot and the target (for locomotion). The symbols v_x, v_y, v_z are the velocity components along the three axes. The symbols *roll* and *pitch* represent the robot's orientation. The symbol (pos_j, vel_j) identifies the position and velocity of the joint j. The symbol $foot\_contact_f$ represents a flag indicating whether the foot f touches the ground ($foot\_contact_f = 1$) or not ($foot\_contact_f = 0$), while cam_s denotes the input from sector s of the camera. Lastly, the symbol m_j indicates the torque applied to joint j.

ID	Sensor	ID	Sensor
0	$z - z_{init}$	6	$roll$
1	$\sin(angle_{to\_target})$	7	$pitch$
2	$\cos(angle_{to\_target})$	8–23	$(pos_j, vel_j) \quad for j \in Num\_joints$
3	$0.3 \times v_x$	24–27	$foot\_contact_f \quad for\ f \in [front\_left/right\_foot, back\_left/right\_foot]$
4	$0.3 \times v_y$	28–33	$cam_s \quad for\ s \in Num\_sectors$
5	$0.3 \times v_z$		
ID	**Motor**		
0–7	$m_j \quad for j \in Num\_joints$		

is iteratively updated through an advanced process consisting of mirrored sampling [76], gradient estimation, and optimization using the Adam optimizer [77]. The algorithm is illustrated in Fig. 5.

In more detail, OpenAI-ES seeks to identify the most promising areas of the solution space (i.e., the connection weights corresponding to better displayed behaviors), so that evolution is targeted toward those regions, increasing the chance to discover effective strategies for the considered problem. To this end, OpenAI-ES evaluates samples in both directions (i.e., mirrored sampling [76]) and ranks them based on fitness. This process aims to reveal the existence of relationships between the weights and the final performance. Lastly, OpenAI-ES performs a gradient estimation based on the fitness ranking and updates the centroid using the Adam optimizer [77], which retains historical data through a pair of

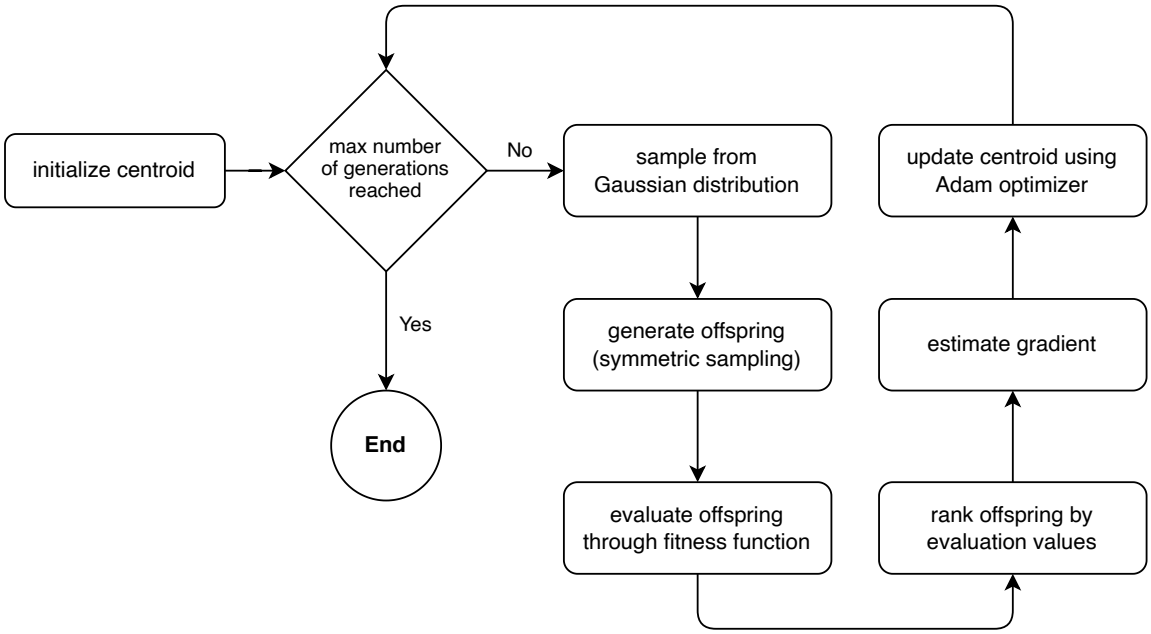


Figure 5: Flow chart of the OpenAI-ES algorithm.

momentum vectors (mean and variance).

The experiments were conducted using *evorobotpy3* [78], a modern simulation tool that contains the implementations of a variety of EAs and some predefined problems. Moreover, it is integrated with libraries such as Gymnasium [79] and Pybullet [49], enabling users to easily create customized simulations.

A detailed list of the parameters used for both scenarios is provided in Table 2.

Table 2
Experimental parameters used in the two considered scenarios. Neural network weights were initialized using the Xavier initialization [80].

Parameter	Simple Aggregation	Aggregation+Locomotion
# of replications	10	10
# of evaluation steps (N_S)	1.5×10^9	10^8
# of samples	20	20
# of robots (N_R)	5	5
# of episodes	3	3
# of episode steps	1000	1000
# of input neurons	12	34
# of hidden neurons	10	20
# of output neurons	4	8
hidden neuron activation function	tanh	tanh
output neuron activation function	tanh	linear
neuron bias	yes	yes
weight initialization	Xavier	Xavier
batch normalization [81]	no	yes

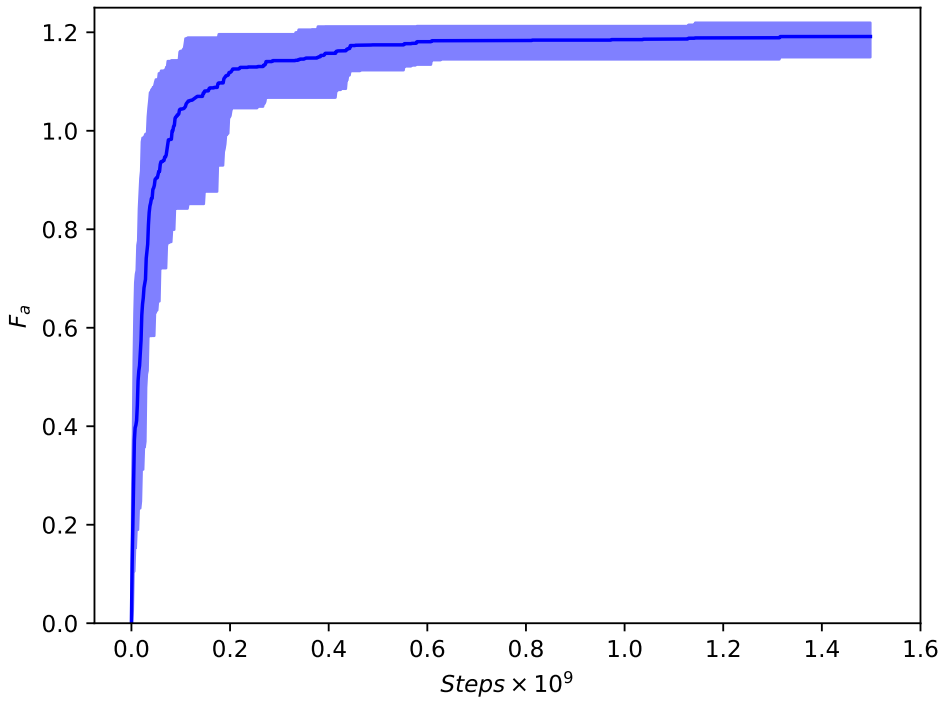
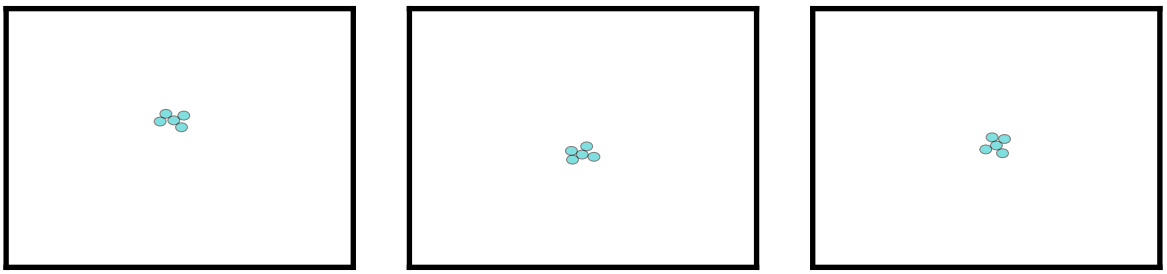


Figure 6: Performance analysis in the Robot Aggregation problem. Shaded area is limited to the range between the first and third quartiles of the data. Data represents the average fitness over 10 replications of the experiment.

4. Results

4.1. Robot Aggregation problem

In the first scenario, we aim to test whether the distance reward D_i, defined in Eq. 3, enables the discovery of effective aggregation behaviors. By examining the performance of the MAS at the end of the evolutionary process, we can see that the average fitness F_a is 1.190 (see Fig. 6), with a standard deviation of 0.028. During the evolutionary process, the analysis of the fitness curve reveals that OpenAI-ES rapidly improves performance in the initial steps and stabilizes at around 5×10^8 steps (see Fig. 6). This implies that the algorithm quickly finds good solutions, whereas the refinement of the discovered strategies requires a longer evolutionary process. If we analyze the final aggregation achieved by the agents, we can observe that the robots manage to discover strategies that ultimately lead to swarm aggregation (see Fig. 7). In particular, the MAS forms a more compact group, located around the center of the arena. This underscores that the distance reward D_i fosters the development of effective aggregation behaviors.



(a) **(b)** **(c)**

Figure 7: Examples of aggregation behaviors discovered with the OpenAI-ES algorithm in the Robot Aggregation problem.

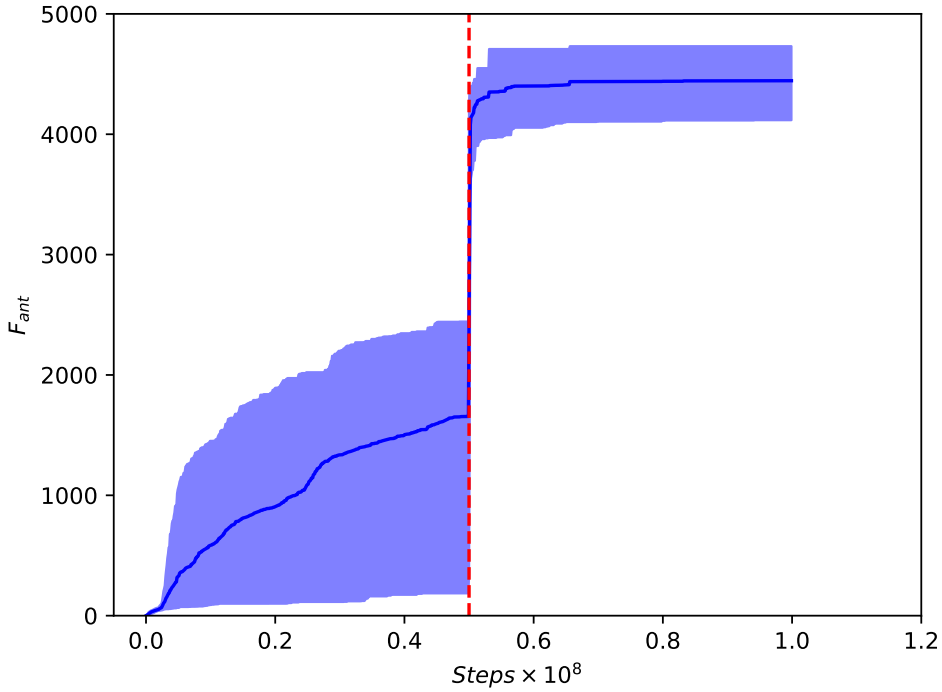


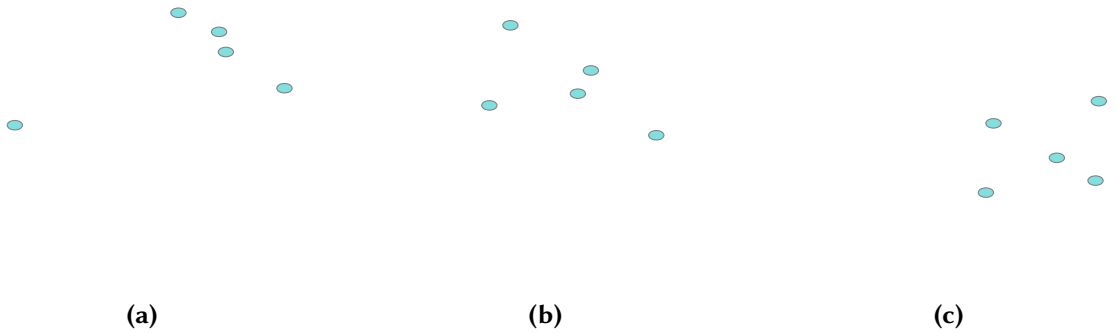
Figure 8: Performance analysis in the AntBullet Swarm problem. Shaded area is limited to the range between the first and third quartiles of the data. Data represents the average fitness over 10 replications of the experiment. The vertical line marks the fitness switch between F_1 (Eq. 6) and F_2 (Eq. 7).

4.2. AntBullet Swarm problem

Based on the considerations reported in the previous section, we investigated whether the fitness function defined in Eq. 5 allows the OpenAI-ES algorithm to evolve strategies in which robots aggregate during locomotion. As we already pointed out in Section 3, we divided the evolutionary process into two separate phases and exploited incremental evolution to enhance agents' performance and capabilities. The mean fitness at the end of evolution is 4444.679 (standard deviation 230.541) and indicates that OpenAI-ES discovers behavioral strategies in which agents exhibit a good locomotion behavior (see the first half of the evolutionary process in Fig. 8). The average fitness obtained before the switch (indicated by the vertical line in Fig. 8) is 1655.366, with a standard deviation of 832.893. These results align with those reported in [47], although the different number of inputs and hidden neurons prevents a direct comparison.

It is worth noting that the ability to locomote is independent of other agents. However, being able to locomote is crucial for the development of aggregation behaviors. In the second phase of the evolutionary process, thanks to the use of the reward function defined in Eq. 3, the agents increase their performance soon after the switch (see the second half of the evolutionary process in Fig. 8), although the improvements in aggregation capability are quite limited. This underscores the difficulty of designing adequate and effective reward functions for MOO problems.

Furthermore, if we examine the final positions of the agents (Fig. 9), we can observe that the aggregation reached by the MAS here is less effective than the previous scenario. In fact, agents fail to move close to one another. Sometimes, the majority of agents succeed in approaching each other (Fig. 9-(a)), while in other cases the MAS disperses (Fig. 9-(b)). Finally, in most cases, the final configuration of the MAS looks similar to the initial cross formation (Fig. 9-(c)). Overall, the modified fitness function and the added camera sensor, useful to perceive the others, allow slight improvements compared to the results reported in [47], where the authors underline the complete absence of such a capability. Nonetheless, the function defined in Eq. 3 does not lead to further enhancements with respect to the aggregation capability.



(a) **(b)** **(c)**

Figure 9: Examples of aggregation behaviors discovered with the OpenAI-ES algorithm in the AntBullet Swarm problem.

To reinforce the aggregation analysis, we calculated the dispersion metric (see [82]), which assesses swarm cohesion and is defined in Eq. 9. This metric was introduced in our previous studies ([14, 15]:

$$d^t = \frac{1}{4{R_R}^2} \sum_{i=1}^{N_R} \|p_i - \bar{p}\|^2 \tag{9}$$

Eq. 9 takes into account the final spatial arrangement of the whole group, where p_i denotes the position of agent i, $\bar{p}$ indicates the COG of the swarm, R_R refers to the robot radius. The notation $\|\cdot\|$ denotes the Euclidean norm.

Thus, to analyze the dynamics of the solutions, Fig. 10 illustrates how dispersion varies during the evaluation of the swarm. As can be seen, the dispersion generally increases on average, reaching a final value of 222.868. This implies that the swarm is dispersing, as the agents have a higher propensity to locomote and are unable to aggregate properly. Examining the dispersion values achieved by the swarm reported in Fig. 11, we observe that the group tends to increase its cohesion, with final dispersion values of 85.621 (Fig. 11-(a)) and 97.025 (Fig. 11-(b)). Therefore, the MOO problem can be addressed more effectively in the best cases, with the agents capable of achieving a higher level of aggregation.

Interestingly, the adoption of the incremental evolution paradigm improves locomotion. Specifically, some agents exhibit refined gaits that make use of all their legs to move (see behavior at https://youtu.be/gW_LdjBOIbs). This allows overcoming the local minima reported in [47], where at least one leg remained extended to maintain stability by avoiding falling. Another discovered locomotion strategy resembles a horse's gallop (see behavior at https://youtu.be/MRquB4HhEFo); indeed, moving quickly clearly helps to maximize the P_i component in Eqs. 6 - 7. However, this type of locomotion conflicts with the objective of aggregation with others, as the rapid movement tends to reduce coordination among agents.

5. Conclusions

In a Multi-Agent System (MAS), agents interact with other entities in order to solve problems that may be very difficult, if not impossible, for a single agent to address. While solving problems consisting of a single objective might be relatively trivial for a MAS, dealing with Multi-Objective Optimization (MOO) is more challenging. It requires the design of appropriate fitness functions and/or the usage of suitable methods in order to make agents able to evolve effective behaviors. In fact, MOO is characterized by conflicting objectives that should be optimized simultaneously, which requires the identification of compromise solutions that perform well across all objectives. For example, evolving both locomotion and aggregation behaviors is particularly challenging, as demonstrated in [47].

In this work, we delve into the analysis of methods that allow the evolution of aggregation in a robotic MAS, particularly focusing on how conflicting objectives can interfere with the evolution of collective behaviors. In more detail, we design a new function D_i (Eq. 3), specifically tailored to promote aggregation, and we test it in two different scenarios. The first one is a Mono-Objective

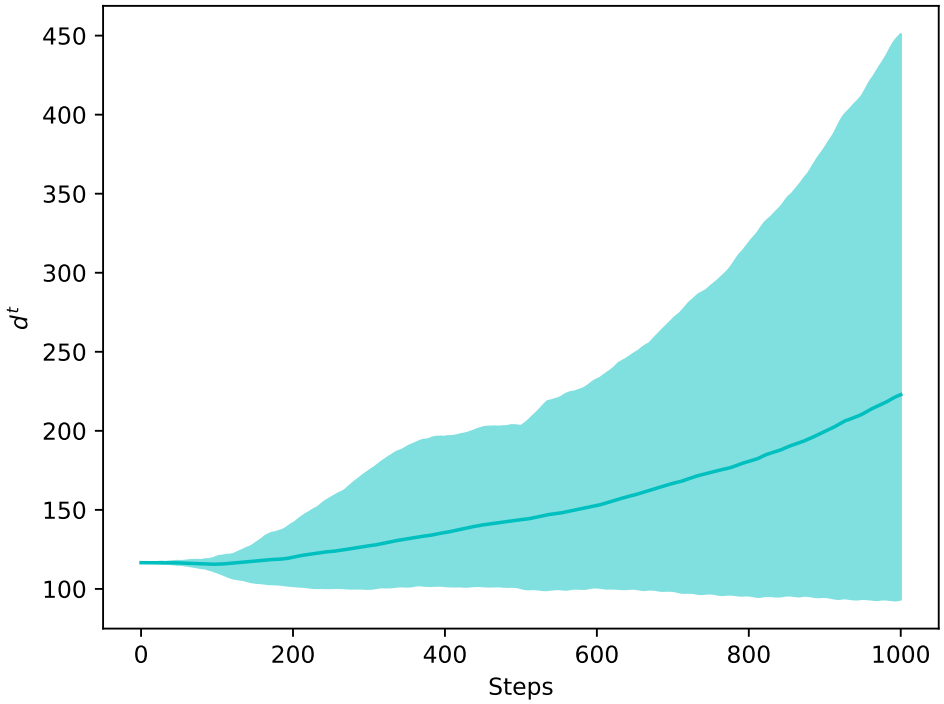
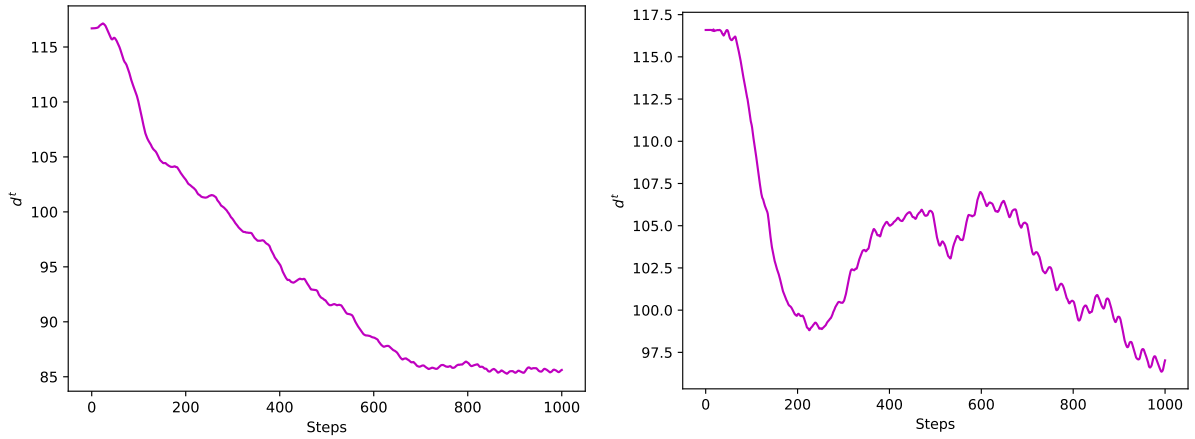


Figure 10: Average dispersion of the swarm over the evaluation process in the AntBullet Swarm problem. Dispersion is computed using the metric defined in Eq. 9, and results are averaged over 10 experimental repetitions.



(a) **(b)**

Figure 11: Dispersion values over time for two of the best-performing cases in terms of swarm cohesion in the AntBullet Swarm problem. These cases exhibit the lowest final dispersion scores, indicating successful aggregation behavior.

Optimization problem where a MAS of 5 MarXbot robots must develop an aggregation capability. The second scenario involves a MOO problem, the AntBullet Swarm problem introduced in [47], in which 5 AntBullet robots must evolve the ability to aggregate while locomoting. In order to foster the development of aggregation behaviors in the latter scenario, we adopted an incremental evolution framework by splitting evolution into two distinct phases. In the first phase, agents must evolve only a locomotion capabilities, which are mandatory to aggregate with others. Instead, in the second phase, agents must evolve both aggregation and locomotion simultaneously. For this purpose, we modified the fitness function defined in [47] by using the new function introduced here and endowing agents with an omni-directional camera that allows them to perceive others. The results indicate

that the function D_i successfully promotes aggregation in the Mono-Objective Optimization scenario. Moreover, it slightly improves the aggregation outcomes with respect to [47], although the agents fail in discovering behavioral strategies that optimize both objectives (i.e., aggregation and locomotion). Finally, the modifications to the AntBullet Swarm problem allow the evolution of an improved locomotion capabilities, which resemble strategies observed in natural organisms.

For future research, we plan to further investigate the design of functions that enable the evolution of effective aggregation behavioral strategies in the AntBullet Swarm problem, as well as the adoption of different frameworks, like ontogenetic approaches [83], to optimize the agent's neural network controller. In this respect, using learning methods, like back-propagation [84] or Spike-Timing-Dependent Plasticity (STDP) [85, 86], could promote the differentiation and/or specialization of agents, ultimately leading to better aggregation capabilities. In addition, employing groups of heterogeneous agents [22] may be valuable for promoting diversity within the MAS. Moreover, more studies will focus on all aspects that could affect the generalizability of learned behaviors. Investigating how performance varies in response to changes in initial conditions, sensory inputs or parameters will reinforce the validity of our approach.

Acknowledgments

The work of A.V. was supported by the National Recovery and Resilience Plan (PNRR)-Ministry of University and Research (MUR) Project through FAIR–Future Artificial Intelligence Research under Grant PE0000013-CUP B53D22000980006.

Declaration on Generative AI

The author(s) have not employed any Generative AI tools.

References

[1] A. Dorri, S. S. Kanhere, R. Jurdak, Multi-agent systems: A survey, IEEE Access 6 (2018) 28573–28593.
[2] V. Julian, V. Botti, Multi-agent systems, Applied Sciences 9 (2019) 1402.
[3] W. Van der Hoek, M. Wooldridge, Multi-agent systems, Foundations of Artificial Intelligence 3 (2008) 887–928.
[4] F. Aldana-Franco, F. Montes-González, S. Nolfi, The improvement of signal communication for a foraging task using evolutionary robotics, Journal of Applied Research and Technology 22 (2024) 90–101.
[5] B. Calvez, G. Hutzler, Automatic tuning of agent-based models using genetic algorithms, in: International workshop on multi-agent systems and agent-based simulation, Springer, 2005, pp. 41–57.
[6] M. Hiraga, Y. Wei, K. Ohkura, Evolving collective cognition of robotic swarms in the foraging task with poison, in: 2019 IEEE Congress on Evolutionary Computation (CEC), IEEE, 2019, pp. 3205–3212.
[7] P. Pagliuca, S. Nolfi, Robust optimization through neuroevolution, PLOS ONE 14 (2019) 1–27.
[8] P. Pagliuca, D. Inglese, A. Vitanza, Measuring emergent behaviors in a mixed competitive-cooperative environment, International Journal of Computer Information Systems and Industrial Management Applications 15 (2023) 69–86.
[9] P. Pagliuca, Learning and evolution: factors influencing an effective combination, AI 5 (2024) 2393–2432.
[10] P. Pagliuca, M. Favia, S. Livi, A. Vitanza, Conceptualizing evolving interdependence in groups: Insights from the analysis of two-agent systems, in: Proceedings of the 21st International Con-

ference on Distributed Computing in Smart Systems and the Internet of Things (DCOSS-IoT), 2025.
[11] P. Pagliuca, M. Favia, S. Livi, A. Vitanza, Interdipendenza nei gruppi: esperimenti con robot sociali, Sistemi intelligenti 2 (2025) 335–355.
[12] E. Bahgeçi, E. Sahin, Evolving aggregation behaviors for swarm robotic systems: A systematic case study, in: Proc. IEEE Swarm Intelligence Symposium, 2005, pp. 333–340.
[13] M. Hiraga, Y. Wei, K. Ohkura, Evolving collective cognition for object identification in foraging robotic swarms, Artificial Life and Robotics 26 (2021) 21–28.
[14] P. Pagliuca, A. Vitanza, Self-organized aggregation in group of robots with OpenAI-ES, in: Int. Conf. on Soft Computing and Pattern Recognition, Springer, 2022, pp. 770–780.
[15] P. Pagliuca, A. Vitanza, Evolving aggregation behaviors in swarms from an evolutionary algorithms point of view, in: Applications of Artificial Intelligence and Neural Systems to Data Science, Springer, 2023, pp. 317–328.
[16] P. Pagliuca, A. Vitanza, A comparative study of evolutionary strategies for aggregation tasks in robot swarms: Macro- and micro-level behavioral analysis, IEEE Access 13 (2025) 72721–72735.
[17] D. H. Stolfi, G. Danoy, Evolutionary swarm formation: From simulations to real world robots, Engineering Applications of Artificial Intelligence 128 (2024) 107501.
[18] V. Trianni, R. Groß, T. H. Labella, E. Şahin, M. Dorigo, Evolving aggregation behaviors in a swarm of robots, in: European Conference on Artificial Life, Springer, 2003, pp. 865–874.
[19] H. M. La, R. S. Lim, W. Sheng, J. Chen, Cooperative flocking and learning in multi-robot systems for predator avoidance, in: 2013 IEEE International Conference on Cyber Technology in Automation, Control and Intelligent Systems, IEEE, 2013, pp. 337–342.
[20] H. M. La, R. Lim, W. Sheng, Multirobot cooperative learning for predator avoidance, IEEE Transactions on Control Systems Technology 23 (2014) 52–63.
[21] J. Li, S. X. Yang, Intelligent collective escape of swarm robots based on a novel fish-inspired self-adaptive approach with neurodynamic models, IEEE Transactions on Industrial Electronics (2024).
[22] P. Pagliuca, A. Vitanza, N-mates evaluation: a new method to improve the performance of genetic algorithms in heterogeneous multi-agent systems., Proceedings of the 24th Edition of the Workshop From Object to Agents (WOA23) 3579 (2023) 123–137.
[23] P. Pagliuca, A. Vitanza, The role of n in the n-mates evaluation method: a quantitative analysis, in: 2024 Artificial Life Conference (ALIFE 2024), MIT press, 2024, pp. 812–814.
[24] M. Brambilla, E. Ferrante, M. Birattari, M. Dorigo, Swarm robotics: a review from the swarm engineering perspective, Swarm Intelligence 7 (2013) 1–41.
[25] E. Şahin, Swarm robotics: From sources of inspiration to domains of application, in: International workshop on swarm robotics, Springer, 2004, pp. 10–20.
[26] H. Hamann, Swarm robotics: A formal approach, volume 221, Springer, 2018.
[27] M. Dorigo, G. Theraulaz, V. Trianni, Swarm robotics: Past, present, and future [point of view], Proceedings of the IEEE 109 (2021) 1152–1165.
[28] N. Horsevad, H. L. Kwa, R. Bouffanais, Beyond bio-inspired robotics: how multi-robot systems can support research on collective animal behavior, Frontiers in Robotics and AI 9 (2022) 865414.
[29] T. Back, Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms, Oxford university press, 1996.
[30] F. J. Gomez, R. Miikkulainen, et al., Solving non-markovian control tasks with neuroevolution, in: Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI), volume 99, 1999, pp. 1356–1361.
[31] C. Igel, Neuroevolution for reinforcement learning using evolution strategies, in: The 2003 Congress on Evolutionary Computation, 2003. CEC'03., volume 4, IEEE, 2003, pp. 2588–2595.
[32] P. Pagliuca, N. Milano, S. Nolfi, Maximizing adaptive power in neuroevolution, PloS one 13 (2018) e0198788.
[33] C. Lamini, S. Benhlima, A. Elbekri, Genetic algorithm based approach for autonomous mobile robot path planning, Procedia Computer Science 127 (2018) 180–189.

[34] P. Pagliuca, S. Nolfi, Integrating learning by experience and demonstration in autonomous robots, Adaptive Behavior 23 (2015) 300–314.
[35] A. Ram, G. Boone, R. Arkin, M. Pearce, Using genetic algorithms to learn reactive control parameters for autonomous robotic navigation, Adaptive behavior 2 (1994) 277–305.
[36] P. Pagliuca, D. Y. Inglese, The importance of functionality over complexity: A preliminary study on feed-forward neural networks, in: Advanced Neural Artificial Intelligence: Theories and Applications, Springer, 2025, pp. 447–458.
[37] S. M. Elsayed, R. A. Sarker, D. L. Essam, A new genetic algorithm for solving optimization problems, Engineering Applications of Artificial Intelligence 27 (2014) 57–69.
[38] P. Pagliuca, Analysis of the exploration-exploitation dilemma in neutral problems with evolutionary algorithms, Journal of Artificial Intelligence and Autonomous Intelligence 1 (2024) 8.
[39] S. Nolfi, D. Floreano, Coevolving predator and prey robots: Do "arms races" arise in artificial evolution?, Artificial life 4 (1998) 311–335.
[40] S. Nolfi, P. Pagliuca, Global progress in competitive co-evolution: a systematic comparison of alternative methods, Frontiers in Robotics and AI 11 (2025) 1470886.
[41] K. Deb, Multi-objective optimisation using evolutionary algorithms: an introduction, in: Multi-objective evolutionary optimisation for product design and manufacturing, Springer, 2011, pp. 3–34.
[42] C. M. Fonseca, P. J. Fleming, An overview of evolutionary algorithms in multiobjective optimization, Evolutionary computation 3 (1995) 1–16.
[43] J. Horn, N. Nafpliotis, D. E. Goldberg, A niched pareto genetic algorithm for multiobjective optimization, in: Proceedings of the first IEEE conference on evolutionary computation. IEEE world congress on computational intelligence, Ieee, 1994, pp. 82–87.
[44] E. Zitzler, Evolutionary algorithms for multiobjective optimization: Methods and applications, volume 63, Shaker Ithaca, 1999.
[45] K. Deb, K. Sindhya, J. Hakanen, Multi-objective optimization, in: Decision sciences, CRC Press, 2016, pp. 161–200.
[46] Y. Censor, Pareto optimality in multiobjective problems, Applied Mathematics and Optimization 4 (1977) 41–59.
[47] P. Pagliuca, A. Vitanza, Enhancing aggregation in locomotor multi-agent systems: a theoretical framework, Proceedings of the 25th Edition of the Workshop From Object to Agents (WOA24) 3735 (2024) 42–57.
[48] T. Salimans, J. Ho, X. Chen, S. Sidor, I. Sutskever, Evolution strategies as a scalable alternative to reinforcement learning, arXiv preprint arXiv:1703.03864 (2017).
[49] E. Coumans, Y. Bai, Pybullet, a python module for physics simulation for games, robotics and machine learning, 2016.
[50] Z. Firat, E. Ferrante, Y. Gillet, E. Tuci, On self-organised aggregation dynamics in swarms of robots with informed robots, Neural Computing and Applications 32 (2020) 13825–13841.
[51] J. Menezes, E. Rangel, B. Moura, Aggregation as an antipredator strategy in the rock-paper-scissors model, Ecological Informatics 69 (2022) 101606.
[52] J. Nauta, P. Simoens, Y. Khaluf, Memory induced aggregation in collective foraging, in: International conference on swarm intelligence, Springer, 2020, pp. 176–189.
[53] T. M. Konijn, K. B. Raper, Cell aggregation in dictyostelium discoideum, Developmental Biology 3 (1961) 725–756.
[54] R. Q. Kidner, E. B. Goldstone, H. J. Rodefeld, L. P. Brokaw, A. M. Gonzalez, N. Ros-Rocher, J. P. Gerdt, Exogenous lipid vesicles induce endocytosis-mediated cellular aggregation in a close unicellular relative of animals, bioRxiv (2024) 2024–05.
[55] S.-H. Cheng, H.-Y. Zhang, M.-Y. Zhu, L. M. Zhou, G.-H. Yi, X.-W. He, J.-Y. Wu, J.-L. Sui, H. Wu, S.-J. Yan, et al., Observations of linear aggregation behavior in rotifers (brachionus calyciflorus), PLoS One 16 (2021) e0256387.
[56] A. Cavagna, A. Cimarelli, I. Giardina, G. Parisi, R. Santagati, F. Stefanini, M. Viale, Scale-free correlations in starling flocks, Proceedings of the National Academy of Sciences 107 (2010)

11865–11870.
[57] B. L. Partridge, T. J. Pitcher, The sensory basis of fish schools: relative roles of lateral line and vision, Journal of comparative physiology 135 (1980) 315–325.
[58] T. J. Pitcher, B. L. Partridge, C. Wardle, A blind fish can school, Science 194 (1976) 963–965.
[59] T. Shibata, T. Fukuda, Coordinative behavior in evolutionary multi-agent system by genetic algorithm, in: IEEE International Conference on Neural Networks, IEEE, 1993, pp. 209–214.
[60] R. O'Grady, R. Groß, A. L. Christensen, M. Dorigo, Self-assembly strategies in a group of autonomous mobile robots, Autonomous Robots 28 (2010) 439–455.
[61] R. Asad, T. Hayakawa, T. Yasuda, Evolutionary design of cooperative transport behavior for a heterogeneous robotic swarm, Journal of Robotics and Mechatronics 35 (2023) 1007–1015.
[62] I. Rechenberg, Evolutionsstrategie: Optimierung technischer systeme nach prinzipien der biologischen evolution, frommann–holzboog, Stuttgart-Bad Cannstatt (1973) 47.
[63] H.-P. Schwefel, Numerische Optimierung von Computer-Modellen mittels der Evolutionsstrategie: mit einer vergleichenden Einführung in die Hill-Climbing-und Zufallsstrategie, volume 1, Springer, 1977.
[64] J. Rais Martínez, F. Aznar Gregori, Comparison of evolutionary strategies for reinforcement learning in a swarm aggregation behaviour, in: Proceedings of the 2020 3rd International Conference on Machine Learning and Machine Intelligence, 2020, pp. 40–45.
[65] N. Hansen, A. Ostermeier, Completely derandomized self-adaptation in evolution strategies, Evolutionary computation 9 (2001) 159–195.
[66] J. H. Holland, Adaptation in natural and artificial systems, University Michigan Press, 1975.
[67] D. Wierstra, T. Schaul, T. Glasmachers, Y. Sun, J. Peters, J. Schmidhuber, Natural evolution strategies, The Journal of Machine Learning Research 15 (2014) 949–980.
[68] Y. Katada, Evolutionary design method of probabilistic finite state machine for swarm robots aggregation, Artificial Life and Robotics 23 (2018) 600–608.
[69] J. Kennedy, R. Eberhart, Particle swarm optimization, in: Proceedings of ICNN'95-international conference on neural networks, volume 4, ieee, 1995, pp. 1942–1948.
[70] A. Leccese, A. Gasparri, A. Priolo, G. Oriolo, G. Ulivi, A swarm aggregation algorithm based on local interaction with actuator saturations and integrated obstacle avoidance, in: 2013 IEEE International Conference on Robotics and Automation, IEEE, 2013, pp. 1865–1870.
[71] M. Bonani, V. Longchamp, S. Magnenat, P. Rétornaz, D. Burnier, G. Roulet, F. Vaussard, H. Bleuler, F. Mondada, The marxbot, a miniature mobile robot opening new perspectives for the collective-robotic research, in: 2010 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, IEEE, 2010, pp. 4187–4193.
[72] F. Gomez, R. Miikkulainen, Incremental evolution of complex general behavior, Adaptive Behavior 5 (1997) 317–342.
[73] J. Kober, J. A. Bagnell, J. Peters, Reinforcement learning in robotics: A survey, The International Journal of Robotics Research 32 (2013) 1238–1274.
[74] P. Pagliuca, N. Milano, S. Nolfi, Efficacy of modern neuro-evolutionary strategies for continuous control optimization, Frontiers in Robotics and AI 7 (2020) 98.
[75] P. Pagliuca, S. Nolfi, The dynamic of body and brain co-evolution, Adaptive Behavior 30 (2022) 245–255.
[76] D. Brockhoff, A. Auger, N. Hansen, D. V. Arnold, T. Hohm, Mirrored sampling and sequential selection for evolution strategies, in: International Conference on Parallel Problem Solving from Nature, Springer, 2010, pp. 11–21.
[77] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, preprint arXiv:1412.6980 (2014).
[78] P. Pagliuca, S. Nolfi, A. Vitanza, Evorobotpy3: a flexible and easy-to-use simulation tool for evolutionary robotics, in: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2025 Companion), 2025.
[79] M. Towers, A. Kwiatkowski, J. Terry, J. U. Balis, G. De Cola, T. Deleu, M. Goulao, A. Kallinteris, M. Krimmel, A. KG, et al., Gymnasium: A standard interface for reinforcement learning environments, preprint arXiv:2407.17032 (2024).

[80] X. Glorot, Y. Bengio, Understanding the difficulty of training deep feedforward neural networks, in: Proceedings of the thirteenth international conference on artificial intelligence and statistics, JMLR Workshop and Conference Proceedings, 2010, pp. 249–256.
[81] S. Ioffe, C. Szegedy, Batch normalization: Accelerating deep network training by reducing internal covariate shift, arXiv preprint arXiv:1502.03167 (2015).
[82] M. Gauci, J. Chen, W. Li, T. J. Dodd, R. Groß, Self-organized aggregation without computation, The Int. Jrnl. of Robotics Research 33 (2014) 1145–1161.
[83] D. Floreano, P. Husbands, S. Nolfi, Evolutionary robotics, Handbook of robotics (2008).
[84] D. E. Rumelhart, G. E. Hinton, R. J. Williams, Learning representations by back-propagating errors, Nature 323 (1986) 533–536.
[85] S. Song, K. D. Miller, L. F. Abbott, Competitive hebbian learning through spike-timing-dependent synaptic plasticity, Nature neuroscience 3 (2000) 919–926.
[86] A. Vitanza, L. Patané, P. Arena, Spiking neural controllers in multi-agent competitive systems for adaptive targeted motor learning, Journal of the Franklin Institute 352 (2015) 3122–3143.