Enhancing Aggregation in Locomotor Multi-Agent Systems: a Theoretical Framework

Paolo Pagliuca[1,*,†], Alessandra Vitanza[1,†]

[1]Institute of Cognitive Sciences and Technologies, National Research Council (CNR-ISTC), Rome, Italy

Abstract

The synthesis of collective behaviors in Multi-Agent Systems is typically approached using various methods, with Evolutionary Algorithms being among the most prevalent. In these systems, agents engage in local interactions with their peers and collectively adopt strategies that manifest at a group level, resembling social behaviors seen in animal societies. We extended the AntBullet problem, which is part of the PyBullet simulation tool, to a collective scenario involving a group of five homogeneous robots to aggregate during locomotion. To evolve this behavior, we employed the OpenAI-ES algorithm alongside a multi-objective fitness function. Our findings indicate that while the robots developed successful locomotion behaviors, they did not exhibit aggregation. This discrepancy is attributed to design choices that unintentionally emphasized locomotion over aggregation capabilities. We discuss the dynamic interplay induced by the fitness function to validate our results and outline future directions. Ultimately, our goal is a first attempt to establish a framework for analyzing collective behaviors using advanced algorithms within modern simulation environments.

Keywords

Multi-Agent Systems, aggregation, fitness function, OpenAI-ES, PyBullet

1. Introduction

In Multi-Agent Systems (MASs) [1], various approaches based on model-free machine learning techniques are employed to address the emergence of collective behaviors. Reinforcement Learning (RL) and Evolutionary Algorithms (EAs) are among the most widespread methods used for this purpose. Common characteristics of such systems include the exploitation of local interactions between agents to identify a common strategy beneficial at the group level, mimicking behaviors observed in social animals such as ants, bees, birds, and fish. Recently, in the simulation panorama, the PyBullet simulation tool [2] has become a standard for customizing environments and testing algorithms like Evolutionary Strategies (ESs) and Reinforcement Learning. Indeed, PyBullet offers a wide range of problems, from classic control tasks to Atari games and locomotion challenges. In this work, we propose a novel framework to investigate collective behaviors in swarms, focusing on the aggregation problem — a common behavior observed in nature [3–6] — that is utilized for tasks such as foraging, collective motion, and defense against predators. Specifically, we extend the AntBullet problem, a standard benchmark

WOA 2024: 25th Workshop "From Objects to Agents", July 8–10, 2024, Forte di Bard (AO), Italy

*Corresponding author.

†The authors contributed equally.

paolo.pagliuca@istc.cnr.it (P. Pagliuca); alessandra.vitanza@istc.cnr.it (A. Vitanza)

0000-0002-3780-3347 (P. Pagliuca); 0000-0002-7760-8167 (A. Vitanza)

in PyBullet, to a collective scenario involving a group of five homogeneous robots with the ultimate goal to aggregate while locomoting. We define a fitness function that rewards agents for both capabilities and use the OpenAI-ES algorithm to evolve such behavior. Our outcomes indicate that the robots evolve a successful locomotion behavior, but do not aggregate. Truly, the fitness function has been designed in a way that achieving a positive reward for aggregation conflicts with the locomotion reward, which is considerably easier to obtain. Consequently, agents behave as individuals living in the same environment but acting egoistically. However, this work represents a first step towards creating a setup that enables the analysis of collective behaviors using modern algorithms in advanced simulation environments.

After a brief introduction about the common techniques and tools adopted in literature (Subsections 1.1 and 1.2), the remaining part of the manuscript will cover the following: an overview of related works to set the relative background (Section 2), followed by the presentation of the theoretical framework (Section 3) and a description of the experimental setup (Section 4). Subsequently, the obtained results will be presented (Section 5), and finally, the discussion and conclusions will be drawn (Section 6).

1.1. Model-free machine learning methods for MASs

Model-free machine learning methods are techniques used in the field of machine learning, where the learning algorithm does not explicitly create or maintain a model of the underlying data distribution. Instead, these methods focus on learning directly from the data through trial and error, often utilizing feedback signals such as rewards or penalties. Therefore, Reinforcement Learning (RL) is a major example of this approach. In this sense, Evolutionary Strategies (ESs) fall under this category because they use a fitness function to guide the optimization process without explicitly modeling the underlying dynamics of the environment. Truly, Evolutionary Algorithms (EAs), including ESs, can be classified as model-free optimization methods rather than traditional model-free learning methods.

Model-free methods are instrumental in scenarios involving complex, highly dynamic, or poorly understood systems, situations that make it challenging to construct an accurate model. These methods excel in tasks such as robotic control, game playing, and autonomous decision-making in uncertain environments. Specifically, the decentralized approaches intrinsic to model-free learning methods are well-suited for MASs, where agents must adapt and coordinate with limited information. Indeed, these approaches provide a powerful framework for learning adaptive behaviors in MASs, fostering the emergence of collective intelligence and enabling agents to navigate dynamic and complex environments effectively. Definitely, applying model-free machine learning in MASs represents opportunities for developing novel algorithms and techniques tailored to multi-agent settings.

1.2. Physics Engines and Simulation Platforms

Physics engines are essential tools for simulating robots and evaluating their performance across various tasks. There are several physics engines and simulation platforms used for simulating robot scenarios. Among these physics engines, we can mention: *(i)* **ODE (Open Dynamics Engine)** [7], the most famous open-source and high-performance library for simulating physics

dynamics. It focuses on real-time simulation and is capable of multi-agent simulation. Widely used in robotics, games, and simulation environments, it is commonly used in research projects [8–10]. *(ii)* **PhysX** [11], a robust physics engine developed by NVIDIA, known for its accuracy and performance in simulating physical interactions. It supports real-time simulations and GPU acceleration, making it popular in game development and high-fidelity simulations in robotics scenarios [12, 13]. Finally, *(iii)* **Bullet Physics** [14], the open-source physics engine chosen for this work. It includes support for collision detection, and rigid and soft body dynamics, and is suitable for simulating complex robotic scenarios. These physics engines provide a robust foundation for implementing simulation platforms, each offering unique features and capabilities useful for studying swarm robotics scenarios. Notable simulation platforms include:

- **Gazebo** [15] is a powerful open-source 3D robotics simulator widely used in research and development. It is a physics-realistic simulator built on ODE, although it recently supports multiple physics engines (including Bullet and others) and integrates with ROS (Robot Operating System) [16]. Due to its features, it is ideal for simulating complex environments with multiple robots, including swarm robotics.
- **CoppeliaSim (formerly V-REP)** [17] is a versatile robot simulation software. It supports a wide range of robots and environments. The simulator offers four physics engines (i.e., Bullet Physics, ODE, Newton, and Vortex Dynamics) and is frequently used for swarm robotics due to its flexibility and ease of creating complex interactions and behaviors.
- **Webots** [18] is an open-source robot simulation software that allows the modeling, programming, and simulation of mobile robots built on ODE. It is excellent for educational purposes and research in swarm robotics, supporting realistic simulations and prototyping.
- **ARGoS (Autonomous Robots Go Swarming)** [19] is a simulation framework specifically designed for swarm robotics. Its most important features are its high efficiency, capability of simulating large-scale robot groups, and customization of both the robots and the environment. It was primarily designed for research in swarm robotics to facilitate experiments and results acquisition.
- **FARSA (Framework for Autonomous Robotics Simulation Applications)** [20] is a simulation framework specifically designed for studying and developing autonomous robots, with a particular focus on swarm robotics. It is particularly suited for researchers who develop and test algorithms for collective intelligence and autonomous decision-making in robotic systems. The major features of FARSA are its high modularity and extensibility, which allow users to extend and customize the simulation framework to specific needs. FARSA supports the rapid creation of custom robots and simulates complex environments in which the robots operate. FARSA is supported by a community of researchers and developers, contributing to its continuous improvement and the availability of shared resources.

1.2.1. PyBullet and AntBullet problem

PyBullet [2] is a physics simulator built on top of the Bullet physics engine [14]. Developed primarily in C++, it was designed with a focus on robotics manipulation, enabling the simulation of articulated rigid bodies, physical joints, and constraints. It also integrates learning algorithms

and supports a range of features including various sensors, gripper and multi-agent simulation, and lightweight graphics rendering. Despite its advantages, *PyBullet* faces challenges such as long simulation times and complexity in setting up the environment. However, its versatility and open-source nature make it a valuable tool in robotics simulation and research. *PyBullet* is easy to use with zero overhead when integrating a physics engine into a Python program. Moreover, it is tailored to encourage and facilitate the use of constraint-based descriptions to abstract physics for modern robotic learning algorithms. Therefore, it can be used with any machine-learning technique that supports PyTorch [21], SimPy [22], in pure Python or conjunction with CUDA [23].

AntBullet [2] is an environmental simulation implemented in *PyBullet* where an ant robot with 8 joints is simulated (*AntBulletEnv*). It serves as a benchmark for reinforcement-learning simulations [24–28]. The goal is to reach a specified end position in the fewest steps possible. In the context of benchmarking, the AntBullet environment is used to evaluate and compare the performance of various algorithms. Generally, the goal is to optimize the movements of the Ant robot, enabling it to perform effectively in the simulation. Therefore, *AntBullet* requires minimal modifications to enable swarm-based robotics.

2. Related works

The background of the present proposal draws direct inspiration from [26]. In this work, the authors describe an experiment comparing the efficacy of various neuro-evolutionary strategies for continuous control optimization. They discuss different algorithms, including OpenAI-ES [29], CMA-ES (Covariance Matrix Adaptation Evolution Strategy) [30], sNES (Separable Natural Evolutionary Strategy)[31], and xNES (Exponential Natural Evolution Strategy) [31], analyzing their performance across a range of tasks. The experiment utilized a variety of benchmark problems, such as locomotion tasks, Atari games, the double-pole balancing problem [32] and a swarm foraging scenario introduced in [33]. The algorithms were evaluated based on their ability to maximize a predefined reward function. Algorithm performance was assessed by the total number of evaluations required to find a solution.

The OpenAI-ES algorithm consistently outperformed other algorithms across all tested problems, demonstrating robustness to changes in hyper-parameters. Results indicate that the success of this method is attributed to the Adam optimizer [34] rather than the virtual batch normalization technique [29, 35]. Furthermore, the effectiveness of the OpenAI-ES algorithm also in achieving collective decision-making for aggregation tasks is demonstrated by the results of our recent study [36]. In that work, we assessed the efficacy of the method both quantitatively, through a performance analysis, and qualitatively by evaluating the emergent behavior in different environmental multi-agent setups.

Similar analyses were conducted in [37], where two evolutionary algorithms, CMA-ES and xNES, were compared in the context of a group of robots making collective decisions by means of aggregation behaviors. The aim of the study was to determine to what extent the performance and the distribution of the robots was affected by the environmental conditions (i.e., dimensions of the sites), and to evaluate the final aggregation of the swarm.

In particular, regarding the AntBullet problem [26], which represents the starting point of

this study, the OpenAI-ES evolutionary strategy outperforms the PPO (Proximal-Policy Optimization) reinforcement learning algorithm [38], despite encountering some issues with the reward function. Specifically, achieving effective behaviors and optimal performance across all replications requires a small adjustment in the reward function, typically a bonus or punishment of approximately ±0.01. Primarily, the authors recommend that future comparisons between evolutionary and reinforcement learning algorithms should incorporate reward functions specifically tailored to each algorithm class. This emphasis underscores the importance of utilizing appropriate reward functions, as evidenced by the fact that reward functions designed for reinforcement learning may not necessarily be effective for evolutionary strategies and vice versa. Therefore, the study highlights the significant impact of the employed reward function on algorithm performance (for a review of the fitness/reward functions used for Evolutionary Algorithms, see [39]). In this context, Table 1 summarizes some interesting works by comparing the algorithms used, the type of task, the defined fitness function, and the homogeneity of the group. It serves to place our proposal in the context of the state-of-the-art.

Table 1
State-of-the-art. Unintroduced acronyms: SARSA (State-Action-Reward-State-Action), DQN_RLaR (Deep Q-Learning with Reinforcement Learning as a Rehersal), RLA (Reinforcement Learning-based Aggregation), FLDDPG (Federated Learning Deep Deterministic Policy Gradient), STDP (Spike timing dependent plasticity), MOEA/D (MultiObjective Evolutionary Algorithm based on Decomposition), GGA (Generational Genetic Algorithm).

Work	Approach	Algorithm	Task	Fitness	Homogeneous
Iima and Kuroe [40]	RL	SARSA [41]	Path planning	Mono	Yes
Nguyen and Banerjee [42]	RL	DQN_RLaR	Foraging	Mono	Yes
Sadeghi Amjadi et al. [43]	RL	RLA [43]	Aggregation	Mono	Yes
Na et al. [44]	RL	FLDDPG [44]	Navigation	Multi	Yes
Vitanza et al. [45]	RL	STDP [46]	Role Specialization	Mono	Yes
Ordaz-Rivas and Torres-Treviño [47]	EA	MOEA/D [48]	Localization	Multi	Yes
Trianni et al. [49]	EA	GGA [50]	Aggregation	Mono	Yes
Kengyel et al. [51]	EA	Wolfpack-inspired EA [52]	Aggregation	Mono	No
Pagliuca et al. [26]	EA	CMA-ES [30] OpenAI-ES [29] sNES [31] xNES [31]	Foraging	Mono	Yes
Pagliuca and Vitanza [36, 37]	EA	CMA-ES OpenAI-ES xNES	Aggregation	Multi	Yes
Pagliuca and Vitanza [53]	EA	GGA	Foraging Escaping from Predator Aggregation	Mono	No
Pagliuca and Vitanza (this)	EA	OpenAI-ES	Aggregation	Multi	Yes

Algorithm 1 *OpenAI-ES algorithm*

Initialize:
  $\sigma \leftarrow 0.02$
  $\lambda = 20$
  γ, f, $optimizer$, θ_0
for $t \leftarrow 0$ to γ **do**
  for $i \leftarrow 0$ to λ **do**
    sample noise vector: $\epsilon_i \sim \mathcal{N}(0, \mathbb{I})$
    evaluate score: ${s_i}^+ \leftarrow f(\theta_t + \sigma \times \epsilon_i)$
    evaluate score: ${s_i}^- \leftarrow f(\theta_t - \sigma \times \epsilon_i)$
  end for
  compute normalized ranks: $u \leftarrow ranks(s), u_i \in [-0.5, 0.5]$
  estimate gradient: $g_t \leftarrow \frac{1}{\lambda} \sum_{i=1}^{\lambda} u_i \times \epsilon_i$
  $\theta_{t+1} \leftarrow \theta_t + optimizer(g_t)$
end for

3. Evolutionary Strategy for Aggregation

Intending to investigate the possibility of synthesizing a successful aggregation behavior in a swarm of AntBullet robots, we employed the OpenAI-ES algorithm [26, 29, 54] (for a description, see Alg. 1), which has been adopted to evolve aggregation in similar settings [36, 55]. Specifically, we considered a swarm of five homogeneous AntBullet robots, controlled through a feed-forward neural network with 28 inputs, 50 internal units and 8 outputs. The list of inputs and outputs of the controller is reported in Table 2.

As we pointed out in Section 1.2.1, the AntBullet robot aims to locomote towards a target direction. The fitness function used to evolve such behavior is defined in [26] and can be expressed by Eq. 1:

$$F_{ant} = P + 0.01 + S + J \tag{1}$$

where the components P, S and J are computed as follows:

$$P = \|pos_{curr} - pos_{prev}\| \tag{2}$$

$$S = -0.01 * \frac{1}{N_m} \sum_{j=1}^{N_m} {m_j}^2 \tag{3}$$

$$J = -0.1 * N_j \tag{4}$$

In Eqs. 2 - 4, pos_{curr} indicates the current position; pos_{prev} denotes the previous position; $\|\cdot\|$ is the norm operator; N_m represents the number of motors; m_j is the value of the $j-th$ motor; N_j indicates the number of joints at limit.

Starting from F_{ant}, we extended the fitness function to deal with our collective scenario, in which the robots must aggregate while locomoting. In particular, we computed the components

Table 2
List of input and output data to the robot's neural network controller. Concerning the inputs, the symbols are defined as follows: z denotes the z coordinate (i.e., height) of the agent; z_{init} indicates the initial z coordinate; $angle_{to\_target}$ represents the relative angle with the target location; v_x, v_y and v_z indicate the robot velocities along the three axes; *roll* and *pitch* are the roll and pitch of the robot; (pos_j, vel_j) represents the position and velocity of the $j-th$ joint; $foot\_contact_f$ denotes the contact flag of the $f-th$ foot (i.e., 1 if the foot touches the ground, 0 otherwise). For outputs, the symbol m_j represents the motor value applied to the $j-th$ joint.

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

P, S and J for each of the five agents (i.e., P_i, S_i and J_i for the generic $i-th$ agent), and we introduced a new component D rewarding agents for the capability of staying at a target distance (set to 1.5m) from the other mates. Given an agent i, the component can be defined as follows:

$$D_i = e^{-100 * \frac{1}{N-1} \sum_{j=1}^{N-1} |dist_{target} - dist_{ij}|} \tag{5}$$

where N is the number of agents (here $N = 5$), $dist_{target}$ represents the desired target distance between each pair of agents and $dist_{ij}$ denotes the distance between the agents i and j.

From Eq. 5, it is evident that robots very far from their mates will not receive any score. Similarly, collisions between peers (i.e., distance below $dist_{target}$) are discouraged since they prevent agents from locomoting. The D component is intended to foster aggregation in the swarm during the motion. Accordingly, the following equation defines the resulting fitness function for the multi-agent scenario:

$$F = \frac{1}{N} \sum_{i=1}^{N} P_i + D_i + S_i + J_i \tag{6}$$

where the single components refer to the generic agent i. We remove the bonus of 0.01 to prevent local minima behaviors, such as staying still.

4. Experimental Setup

We extended the AntBullet problem [2] to deal with a collective scenario involving a group of 5 robots. The starting locations of the agents remain almost constant, and the initial formation of the swarm is a cross, as shown in Fig. 1. A small amount of noise is added to the joints' initial positions to make the problem stochastic (parameter *input noise* in Table 3).

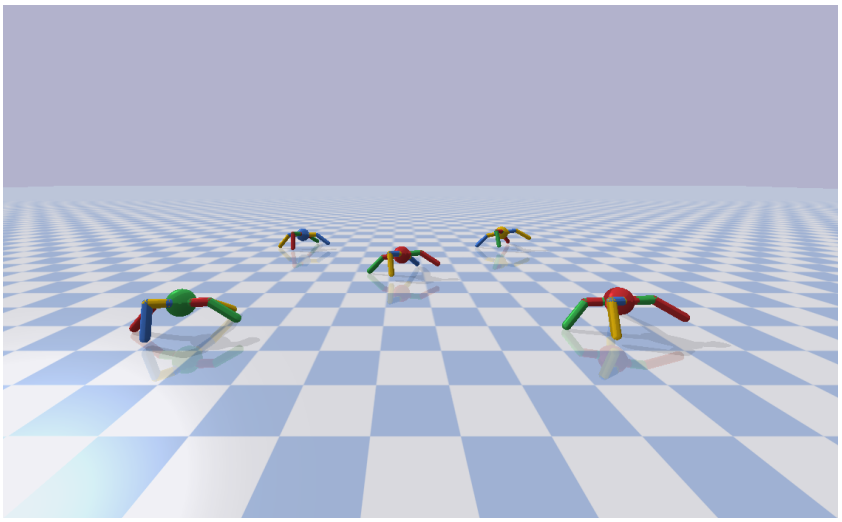

Figure 1: The evolutionary task. The swarm consists of five AntBullet robots. Initially, the agents are placed to form a cross.

The experiment has been repeated 30 times. Evolution lasts 5×10^7 evaluation steps. The ability of the swarm to cope with the problem is estimated by evaluating the group in a single episode lasting up to 1000 steps. Moreover, the generalization capability of the swarm is computed over 3 post-evaluation episodes, each lasting a maximum of 1000 steps. An evaluation episode is prematurely stopped if at least one of the agents falls on the ground. The full list of evolutionary parameters is provided in Table 3.

5. Results and Analysis

In this section, we report the outcomes of our experiments. The average fitness ($AvgF$) obtained in 30 replications of the experiment is 1769.389, with a standard deviation ($StdF$) of 419.417. The best replication achieved a fitness score of 2551.119, while the worst replication obtained a fitness value of 862.517. Overall, 16 out of 30 replications (i.e., 53.33%) got a fitness value over average (see Fig. 2).

If we analyze the impact of the P, D, S and J components of the fitness function F, we observe that the score is mostly due to the first one (see Fig. 3). In fact, progressing toward a target (P) is easier and does not depend on the behavioral capabilities of the mates. Conversely, reducing the distance from the peers (D) — i.e. aggregating — implies moving towards other mates, thus decreasing the reward provided by the P component. However, the D component rewards agents

Table 3
Evolutionary parameters. The symbols γ, λ and σ have the same meaning as indicated in Alg. 1. The U symbol denotes the uniform distribution.

Parameter	Value
# of replications	30
# of evaluation steps (γ)	5×10^7
# of symmetric samples (λ)	20
σ	0.02
# of robots	5
# of episodes	1
# of post-evaluation episodes	3
# of episode steps	1000
# of hidden neurons	50
activation function (hidden neurons)	tanh
activation function (output neurons)	linear
input noise	$\eta_{in} \in U(-0.1, 0.1)$
output noise	$\eta_{out} \in U(-0.01, 0.01)$

only if the mutual distance with their peers is very close to the target distance $dist_{target}$ (see Eq. 5). This explains why it does not play any role in this context. Similarly, the S component has almost no impact on F, that is the OpenAI-ES algorithm synthesizes solutions not applying

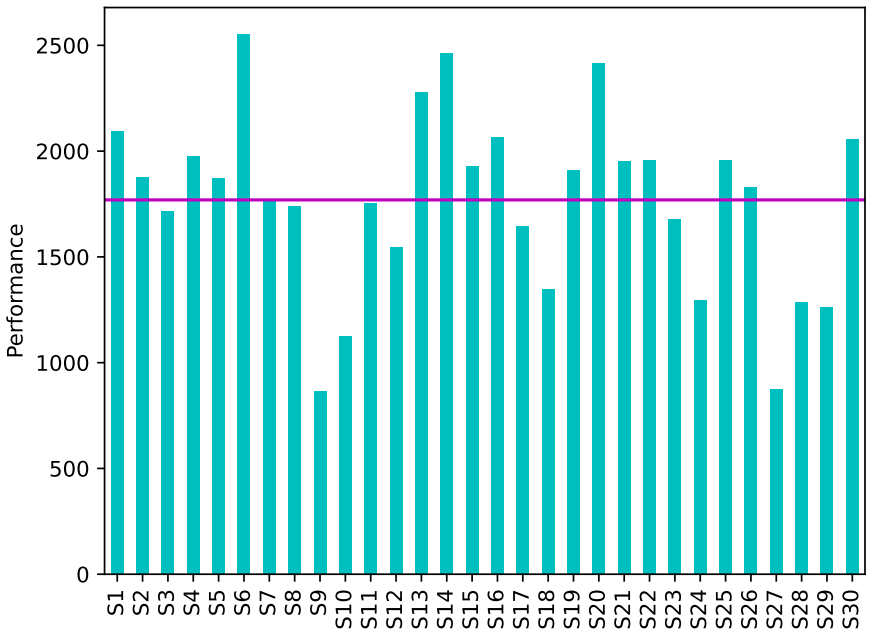

Figure 2: Fitness distribution. Light blue bars represent the performance obtained in each of the 30 replications of the experiment. The horizontal line marks the average fitness achieved.

strong values to the motors. Finally, the J component provides a punishment of around 0.25 per step during the evaluation episode. This means that the AntBullet robots typically push the joints of one of the four legs at their limits. This strategy is helpful to keep balance on the ground (see Fig. 4).

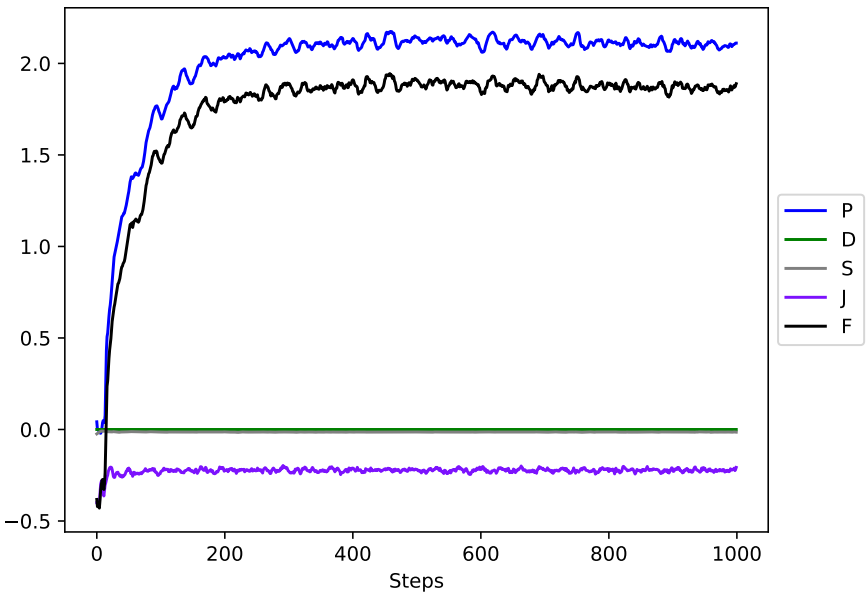

Figure 3: Fitness components (P, D, S and J, see Eqs. 2, 4) and overall fitness F (see Eq. 6). These data have been collected from 30 replications of the experiment.

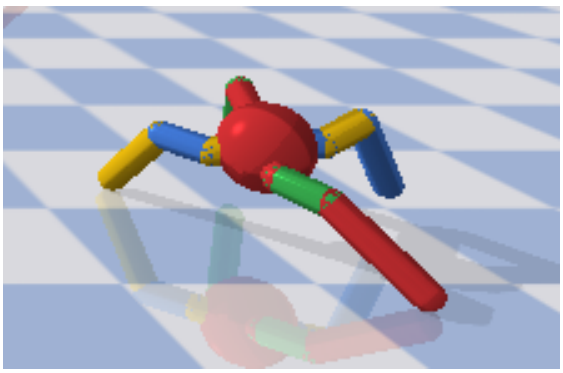

Figure 4: An example of the behavioral strategy of one AntBullet robot: the back right leg is almost fully extended in order to maintain stability on the ground.

Overall, our outcomes demonstrate that the fitness function F makes the robots act egoistically, since the two objectives — locomote and aggregate — conflict with each other. This strategy is spontaneously discovered by the OpenAI-ES algorithm, which attempts to optimize F. A similar finding can be seen in [56], where no global incentive to cooperate or compete is given to the group, and the chosen fitness does not drive agents to a preferred solution. In this case, the

evolutionary strategy evolves agents that implicitly include selfish sub-tasks if the coordination within a team proves complex. Indeed, in this way, they can adapt to situational circumstances and may change their propensity depending on the specific situation.

Moreover, the individualistic inclination of the agents can be further explained by considering that the group is formed by homogeneous robots, which share the same network controller and the same physical embodiment. However, in [53] the authors show how the group dynamics in heterogeneous MASs are influenced by the different skills of the agents, which are more adaptive and behave according to the particular mates they are evaluated with.

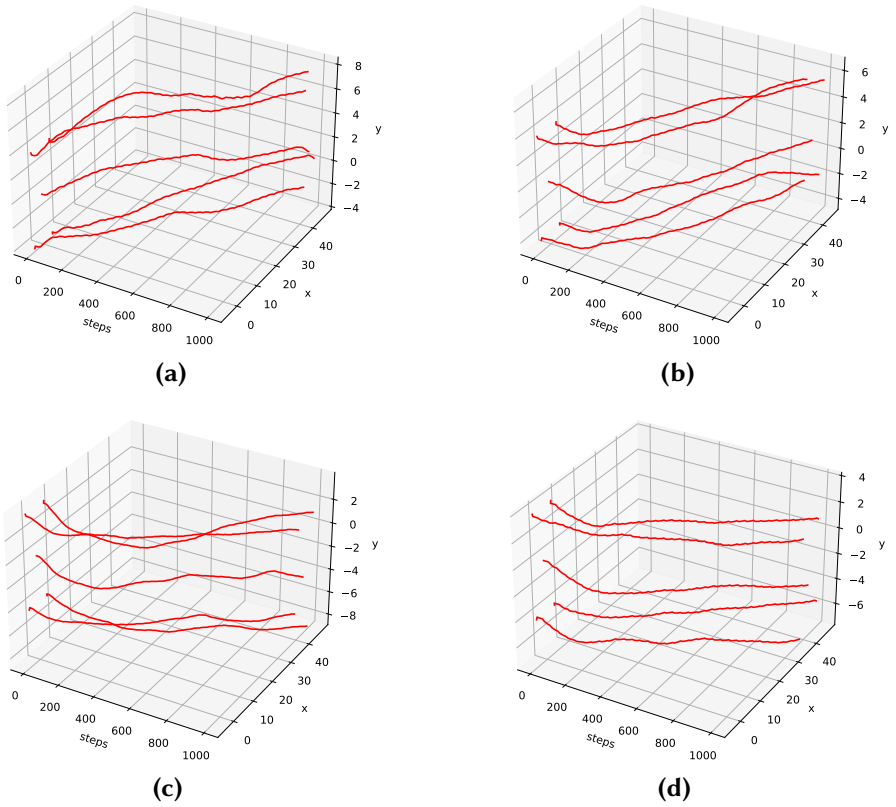

Figure 5: Examples of trajectories performed by the agents of the swarm. The behavioral strategies of the robots evolved in the replications that obtained a fitness score $F > AvgF + StdF$: S6 (a), S13 (b), S14 (c) and S20 (d) (see Fig. 3 for details) were analyzed. The robots usually move by maintaining the initial cross formation throughout the evaluation episode (e.g., (a), (b), (d)). In some cases, the pressure to stay closer emerges during the motion, with some of the agents moving close and even crossing their paths (e.g., (c)). Nevertheless, the overall mutual distance between the group's members is not sufficient to get a score from the D component (see Fig. 3). This explains why the P component of the fitness function plays a more relevant role than the D component.

Fig. 5 shows the behavioral strategies exhibited by the agents evolved in the replications where $F > AvgF + StdF$. Data are collected in a post-evaluation episode. As can be seen, the agents generally move towards the target regardless of the behavior of their mates. The

pressure to aggregate does not emerge throughout the episode, although some of the robots may sometimes cross their paths (see Fig. 5, c).

To summarize, the results reveal the fitness function's importance on the swarm's evolved behavior. Indeed, the agents successfully tackle the evolutionary problem (i.e., they manage to locomote towards a target destination), though the aggregation is not reached. Carefully designing the fitness function to address all the objectives is paramount for enhancing a successful collective strategy.

6. Discussion and Conclusion

In this work, we proposed a theoretical framework to analyze collective behaviors in MASs by exploiting modern simulation environments and using state-of-the-art Evolutionary Algorithms. In particular, we extended the benchmark AntBullet problem to a collective scenario involving a swarm of five AntBullet robots, whose goal was to aggregate during locomotion. We designed a multi-objective fitness function rewarding agents for both capabilities. The results we achieved demonstrate that agents successfully developed a locomotion capability, but they did not aggregate. Undoubtedly, the definition of a multi-objective fitness function presents several challenges because, in many cases, the objectives conflict with each other. Essentially, improving performance on one objective might degrade performance on another. This was specifically the case where the attempt to optimize the D component might have degraded the P component. This contrast made it difficult to find an optimal solution that satisfied all objectives simultaneously. Moreover, in this case, a problem with scaling may have occurred. The different components might have had different scales during the trials, especially because, in dynamic environments, the importance of different objectives might change over time. This implied that one objective might have dominated the others, leading to biased results. This aligns with the recommendation from the aforementioned study [26], emphasizing the importance of utilizing appropriate reward functions tailored to each task. Thus, addressing these challenges often requires careful consideration of the specific characteristics of the problem at hand.

Regarding future research directions, several ideas emerge. Undoubtedly, future work should focus on refining the multi-objective fitness function to better balance conflicting sub-goals. Additionally, investigating alternative state-of-the-art evolutionary algorithms could produce more sophisticated strategies for achieving both locomotion and aggregation. Comparing these alternatives with OpenAI-ES, as done in some of our works, might provide insights into which algorithms are more suitable for multi-objective tasks in dynamic environments. Moreover, introducing communication mechanisms among the agents could potentially enhance the emergence of aggregation behavior. Understanding how communication affects the swarm's ability to aggregate while maintaining effective locomotion could lead to developing more sophisticated strategies. Lastly, as we discussed in Section 5, using a heterogeneous MAS similar to [53] might foster the emergence of more complex dynamics, in which the individual skills may play different roles and the resulting overall behavior cannot be predicted a prior. In conjunction with this, efficient communication requires strong collaboration, and robots need more intelligence to handle unknown situations. These demands are challenging to meet with current architectures and hardware. Creating communication networks between heterogeneous

agents and designing efficient communication protocols are key challenges that need further exploration. Indeed, considering hybrid solutions within the context of Edge Intelligence platforms (EIP) or the Internet of Things (IoT), combined with evolutionary strategies for agent populations, could be a promising direction for future research.

Acknowledgement

A.V. acknowledges support from the PNRR MUR project PE0000013-FAIR - Future Artificial Intelligence Research.

References

[1] P. G. Balaji, D. Srinivasan, An Introduction to Multi-Agent Systems, Springer Berlin Heidelberg, Berlin, Heidelberg, 2010, pp. 1–27.

[2] E. Coumans, Y. Bai, Pybullet, a python module for physics simulation for games, robotics and machine learning (2016).

[3] J.-L. Deneubourg, A. Lioni, C. Detrain, Dynamics of aggregation and emergence of cooperation, The Biological Bulletin 202 (2002) 262–267.

[4] S. Garnier, C. Jost, R. Jeanson, J. Gautrais, M. Asadpour, G. Caprari, G. Theraulaz, Aggregation behaviour as a source of collective decision in a group of cockroach-like-robots, in: European conference on artificial life, Springer, 2005, pp. 169–178.

[5] R. Jeanson, C. Rivault, J.-L. Deneubourg, S. Blanco, R. Fournier, C. Jost, G. Theraulaz, Self-organized aggregation in cockroaches, Animal behaviour 69 (2005) 169–180.

[6] B. Oldroyd, A. Smolenski, S. Lawler, A. Estoup, R. Crozier, Colony aggregations in apis mellifera l, Apidologie 26 (1995) 119–130.

[7] R. Smith, Open dynamics engine, 2008. URL: http://www.ode.org/.

[8] V. Ondroušek, The Solution of 3D Indoor Simulation of Mobile Robots Using ODE, Springer Berlin Heidelberg, Berlin, Heidelberg, 2010, pp. 215–220. doi:10.1007/978-3-642-05022-0_37.

[9] P. Arena, L. Patané, P. S. Termini, A. Vitanza, R. Strauss, Software/hardware issues in modelling insect brain architecture, in: S. Jeschke, H. Liu, D. Schilberg (Eds.), Intelligent Robotics and Applications, Springer Berlin Heidelberg, Berlin, Heidelberg, 2011, pp. 46–55.

[10] A robotic simulation framework for cognitive systems, Spatial Temporal Patterns for Action-Oriented Perception in Roving Robots II: An Insect Brain Computational Model (2014) 153–176.

[11] NVIDIA, Physx library, 2008. URL: https://developer.nvidia.com/physx-sdk.

[12] J. Lächele, A. Franchi, H. H. Bülthoff, P. Robuffo Giordano, Swarmsimx: Real-time simulation environment for multi-robot systems, in: I. Noda, N. Ando, D. Brugali, J. J. Kuffner (Eds.), Simulation, Modeling, and Programming for Autonomous Robots, Springer Berlin Heidelberg, Berlin, Heidelberg, 2012, pp. 375–387.

[13] M. Mittal, C. Yu, Q. Yu, J. Liu, N. Rudin, D. Hoeller, J. L. Yuan, R. Singh, Y. Guo, H. Mazhar, A. Mandlekar, B. Babich, G. State, M. Hutter, A. Garg, Orbit: A unified simulation frame-

work for interactive robot learning environments, IEEE Robotics and Automation Letters 8 (2023) 3740–3747. doi:10.1109/LRA.2023.3270034.

[14] E. Coumans, Bullet physics simulation, in: ACM SIGGRAPH 2015 Courses, SIGGRAPH '15, Association for Computing Machinery, New York, NY, USA, 2015. doi:10.1145/2776880.2792704.

[15] N. Koenig, A. Howard, Design and use paradigms for Gazebo, an open-source multi-robot simulator, in: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566), volume 3, 2004, pp. 2149–2154.

[16] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, A. Y. Ng, et al., Ros: an open-source robot operating system, in: ICRA workshop on open source software, volume 3, Kobe, Japan, 2009, p. 5.

[17] E. Rohmer, S. P. N. Singh, M. Freese, Coppeliasim (formerly v-rep): a versatile and scalable robot simulation framework, in: Proc. of The International Conference on Intelligent Robots and Systems (IROS), 2013.

[18] Webots, http://www.cyberbotics.com, ???? URL: http://www.cyberbotics.com, open-source Mobile Robot Simulation Software.

[19] C. Pinciroli, V. Trianni, R. O'Grady, G. Pini, A. Brutschy, M. Brambilla, N. Mathews, E. Ferrante, G. Di Caro, F. Ducatelle, M. Birattari, L. M. Gambardella, M. Dorigo, Argos: a modular, parallel, multi-engine simulator for multi-robot systems, Swarm intelligence 6 (2012) 271–295.

[20] G. Massera, T. Ferrauto, O. Gigliotta, S. Nolfi, FARSA: An Open Software Tool for Embodied Cognitive Science, in: Proceedings of the 12th European Conference on Artificial Life (ECAL 2013), 2013, pp. 538–545.

[21] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, S. Chintala, Pytorch: An imperative style, high-performance deep learning library, 2019. arXiv:1912.01703.

[22] A. Meurer, C. P. Smith, M. Paprocki, O. Čertík, S. B. Kirpichev, M. Rocklin, A. Kumar, S. Ivanov, J. K. Moore, S. Singh, T. Rathnayake, S. Vig, B. E. Granger, R. P. Muller, F. Bonazzi, H. Gupta, S. Vats, F. Johansson, F. Pedregosa, M. J. Curry, A. R. Terrel, v. Roučka, A. Saboo, I. Fernando, S. Kulal, R. Cimrman, A. Scopatz, Sympy: symbolic computing in python, PeerJ Computer Science 3 (2017) e103. doi:10.7717/peerj-cs.103.

[23] S. Cook, CUDA Programming: A Developer's Guide to Parallel Computing with GPUs, 1st ed., Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2012.

[24] R. Akrour, D. Tateo, J. Peters, Continuous action reinforcement learning from a mixture of interpretable experts, IEEE Transactions on Pattern Analysis and Machine Intelligence 44 (2021) 6795–6806.

[25] S. Dankwa, W. Zheng, Twin-delayed ddpg: A deep reinforcement learning technique to model a continuous movement of an intelligent robot agent, in: Proceedings of the 3rd international conference on vision, image and signal processing, 2019, pp. 1–5.

[26] P. Pagliuca, N. Milano, S. Nolfi, Efficacy of modern neuro-evolutionary strategies for continuous control optimization, Frontiers in Robotics and AI 7 (2020) 98.

[27] D. Reda, T. Tao, M. van de Panne, Learning to locomote: Understanding how environment design matters for deep reinforcement learning, in: Proceedings of the 13th ACM

SIGGRAPH Conference on Motion, Interaction and Games, 2020, pp. 1–10.
[28] Q. Zhang, L. Zhang, Q. Ma, J. Xue, The lstm-per-td3 algorithm for deep reinforcement learning in continuous control tasks, in: 2023 China Automation Congress (CAC), IEEE, 2023, pp. 671–676.
[29] T. Salimans, J. Ho, X. Chen, S. Sidor, I. Sutskever, Evolution strategies as a scalable alternative to reinforcement learning (2017).
[30] N. Hansen, A. Ostermeier, Completely derandomized self-adaptation in evolution strategies, Evolutionary computation 9 (2001) 159–195.
[31] D. Wierstra, T. Schaul, T. Glasmachers, Y. Sun, J. Peters, J. Schmidhuber, Natural evolution strategies, The Jrnl of Machine Learning Research 15 (2014).
[32] A. P. Wieland, Evolving neural network controllers for unstable systems, in: IJCNN-91-Seattle International Joint Conference on Neural Networks, volume 2, IEEE, 1991, pp. 667–673.
[33] P. Pagliuca, S. Nolfi, Robust optimization through neuroevolution, PLOS ONE 14 (2019) 1–27.
[34] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, arXiv:1412.6980 (2014).
[35] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, X. Chen, Improved techniques for training gans, Advances in neural information processing systems 29 (2016).
[36] 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.
[37] 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.
[38] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, O. Klimov, Proximal policy optimization algorithms, arXiv preprint arXiv:1707.06347 (2017).
[39] A. L. Nelson, G. J. Barlow, L. Doitsidis, Fitness functions in evolutionary robotics: A survey and analysis, Robotics and Autonomous Systems 57 (2009) 345–370.
[40] H. Iima, Y. Kuroe, Swarm reinforcement learning algorithms based on sarsa method, in: 2008 SICE Annual Conference, IEEE, 2008, pp. 2045–2049.
[41] R. S. Sutton, A. G. Barto, Temporal-difference learning (1998).
[42] T. Nguyen, B. Banerjee, Reinforcement learning as a rehearsal for swarm foraging, Swarm Intelligence 16 (2022) 29–58.
[43] A. Sadeghi Amjadi, C. Bilaloğlu, A. E. Turgut, S. Na, E. Şahin, T. Krajník, F. Arvin, Reinforcement learning-based aggregation for robot swarms, Adaptive Behavior 32 (2024) 265–281.
[44] S. Na, T. Rouček, J. Ulrich, J. Pikman, T. Krajník, B. Lennox, F. Arvin, Federated reinforcement learning for collective navigation of robotic swarms, IEEE Transactions on cognitive and developmental systems 15 (2023) 2122–2131.
[45] 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. URL: https://www.sciencedirect.com/science/article/pii/S001600321500174X. doi:https://doi.org/10.1016/j.jfranklin.2015.04.014, special Issue on Advances in Nonlinear Dynamics and Control.

[46] S. Song, K. D. Miller, L. F. Abbott, Competitive hebbian learning through spike-timing-dependent synaptic plasticity, Nat Neurosci 3 (2000) 919–926.
[47] E. Ordaz-Rivas, L. Torres-Treviño, Improving performance in swarm robots using multi-objective optimization, Mathematics and Computers in Simulation 223 (2024) 433–457.
[48] Q. Zhang, H. Li, Moea/d: A multiobjective evolutionary algorithm based on decomposition, IEEE Transactions on evolutionary computation 11 (2007) 712–731.
[49] V. Trianni, R. Groß, T. H. Labella, E. Şahin, M. Dorigo, Evolving aggregation behaviors in a swarm of robots, in: Advances in Artificial Life: 7th European Conference, ECAL 2003, Dortmund, Germany, September 14-17, 2003. Proceedings 7, Springer, 2003, pp. 865–874.
[50] J. J. Grefenstette, et al., Genetic algorithms for changing environments, in: Ppsn, volume 2, Citeseer, 1992, pp. 137–144.
[51] D. Kengyel, H. Hamann, P. Zahadat, G. Radspieler, F. Wotawa, T. Schmickl, Potential of heterogeneity in collective behaviors: A case study on heterogeneous swarms, in: PRIMA 2015: Principles and Practice of Multi-Agent Systems: 18th International Conference, Bertinoro, Italy, October 26-30, 2015, Proceedings 13, Springer, 2015, pp. 201–217.
[52] P. Zahadat, T. Schmickl, Wolfpack-inspired evolutionary algorithm and a reaction-diffusion-based controller are used for pattern formation., in: GECCO, 2014, pp. 241–248.
[53] 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.
[54] P. Pagliuca, S. Nolfi, The dynamic of body and brain co-evolution, Adaptive Behavior 30 (2022) 245–255.
[55] 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.
[56] 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.