Foraging Swarms using Multi-Agent Reinforcement Learning

Carsten Hahn, Fabian Ritz, Paula Wikidal, Thomy Phan, Thomas Gabor and Claudia Linnhoff-Popien

Mobile and Distributed Systems Group, LMU Munich, Munich, Germany
carsten.hahn@ifi.lmu.de

Abstract

Flocking or swarm behavior is a widely observed phenomenon in nature. Although the entities might have self-interested goals like evading predators or foraging, they group themselves together because a collaborative observation is superior to the observation of a single individual. In this paper, we evaluate the emergence of swarms in a foraging task using multi-agent reinforcement learning (MARL). Every individual can move freely in a continuous space with the objective to follow a moving target object in a partially observable environment. The individuals are self-interested as there is no explicit incentive to collaborate with each other. However, our evaluation shows that these individuals learn to form swarms out of self-interest and learn to orient themselves to each other in order to find the target object even when it is out of sight for most individuals.

Introduction

Swarm behavior is a common phenomenon in nature, where multiple entities are grouping themselves together in order to evade predators, forage, or make collective decisions, for example, selecting a new nest. It has been up to debate if this behavior could have arisen out of pure self-interest on the individual level in evolution, e.g., if fishes might group themselves together to increase their individual survival chances at the cost of some other fish's life or if birds might flock to find a food source much faster despite having to share the food with each other afterwards. However, for an artificial swarm in a simplified predator-prey setting, Hahn et al. (2019) have shown that swarm behavior can emerge purely out of individual self-interest, where each entity aims to increase its own chance of survival.

Simulating this phenomenon is interesting for a variety of areas, where swarm behavior needs to be understood, for example, to plan the location of emergency exits or in rescue scenarios, where multiple helpers have to find and save victims in a decentralized way (see O'Grady et al. (2009), e.g.). Understanding the emergence of swarms resulting from self-interested objectives and partial observability could lead to novel ways of solving real-world problems like traffic efficiency, where multiple drivers navigate through a city, or communication networks, where different network users have to efficiently share a common network resource. In all these settings, there are only self-interested entities with limited information about the global state.

Previous approaches to simulate swarm behavior either used a fixed set of rules or use reward shaping to enforce swarms with an explicit optimization objective. While these approaches can lead to emerging swarms, they require detailed prior knowledge about the application area. Furthermore, these approaches do not provide further insights about swarm behavior since the swarm is artificially enforced.

In this paper, we build on Hahn et al. (2019) to evaluate the emergence of swarms in a foraging task using multi-agent reinforcement learning (MARL). We focus on a setting where multiple autonomous entities have to follow a moving target object in a partially observable environment. All entities are self-interested so that there is no direct incentive to collaborate with each other. In comparison to Hahn et al. (2019) we use positive reinforcement instead of a penalty for being caught. Our evaluation shows simple cooperative behavior, i.e., the individuals learn to form swarms and thus follow the target object even when it is out of sight for most individuals.

Reinforcement Learning

We formulate our problem as a *stochastic game (SG)* $M_{SG} = \langle \mathcal{D}, \mathcal{S}, \mathcal{A}, \mathcal{P}, (\mathcal{R}_e)_{1 \leq e \leq N}, \mathcal{Z}, \Omega \rangle$, where is $\mathcal{D} = \{1, ..., N\}$ is a set of agents, $\mathcal{S}$ is a set of states s, $\mathcal{A} = \mathcal{A}_1 \times ... \times \mathcal{A}_N$ is the set of *joint actions* a, $\mathcal{P}(s'|s, a)$ is the transition probability, $r_e = \mathcal{R}_e(s, a)$ is a scalar reward for agent $e \in \mathcal{D}$, $\mathcal{Z}$ is a set of local observations o_e for each agent $e \in \mathcal{D}$, and $\Omega(s, a) = o' = \langle o'_1, ..., o'_N \rangle \in \mathcal{Z}^N$ is the joint observation function.

The goal of each agent e is to find a *best response* policy $\pi_e : \mathcal{Z} \rightarrow \mathcal{A}_e$ which maximizes its expected return $Q_e^\pi(s, a) = \mathbf{E}_{a=\pi(s)}[G|s, a]$, where $\pi = \langle \pi_1, ..., \pi_N \rangle$ is the joint policy of all agents and $G_t = \sum_{k=1}^{\infty} \gamma^{k-1} \mathcal{R}_e(s_k, a_k)$ is the return with discount factor $\gamma \in [0, 1]$. Note that Q_e always depends on the current policies of other agents.

Best responses can be found with *reinforcement learn-*

ing (RL), where π_e is learned with experience tuples $\langle o_e, a_e, r_e, o'_e \rangle$, which are obtained from agent interaction with the environment. *Q-Learning* is a popular approach to RL, where $\hat{Q}_e(o_e, a_e) \approx Q_e^\pi$ is approximated with the following update rule Watkins and Dayan (1992):

$$\hat{Q}_e(o_e, a_e) \leftarrow (1 - \alpha) \cdot \hat{Q}_e(o_e, a_e) + \alpha \cdot y \qquad (1)$$

where $y = r_e + \gamma \cdot \max_{a'_e} \hat{Q}_e(o'_e, a'_e)$ and $\alpha \in [0, 1]$ is the learning rate.

Current state-of-the-art RL is implemented with deep learning using *Deep Q-Networks (DQN)* and has been applied to multi-agent systems in a variety of domains (Mnih et al. (2015); Leibo et al. (2017); Gupta et al. (2017)).

Emergent Swarming

A fundamental paper of Reynolds (1987) on swarm simulation formulates three basic behavior rules for autonomously acting units (*Boids*) to form a swarm:

Cohesion Navigate to the centered position of the neighboring Boids.

Separation Move away from other Boids to keep a minimum distance.

Alignment Adjust the own alignment to that of the neighboring Boids.

These rules are based solely on local information. Each Boid requires only position and movement direction of its nearest neighbors but does not need an overview of the entire swarm. This approach is highly scalable as even if the number of Boids increases, the amount of information to determine each Boid's movement direction remains the same. Each of these three rules can be understood as a force that deflects the Boids in a certain direction.

Subsequently, Reynolds (1999) developed algorithms to produce natural collective behavior in situations where individuals follow a common leader. This is achieved by combining *separation* as described above and the so-called *arrival* behavior. Essentially, a continuous force is applied between a Boid's current and its target position. This force deflects the Boid's direction of movement until it finally arrives at the target. An additional force adds a decelerating effect. This force is used to slow down the Boid as it approaches the target so that it does not move beyond the target. For a Boid to pursue a leader, the target position is defined with certain distance behind the leader's actual position.

These behaviors have been applied to self-organization in a MARL system by Morihiro et al. (2008). Instead of fixed rules, the agents can choose between four different actions to change their movement direction. While each agent may observe any other agent, the distance to the respective neighbor determines that observation's influence on the actual action. Additionally, a predator appears periodically and the agents are provided with an according perception range. The reward function yields positive reward for keeping a certain distance to the nearest neighbors and moving away from the predator. While in the experiments the agents developed a strong swarm cohesion, this was mainly achieved by shaping the rewards analogous to Reynold's behavior rules. Furthermore, no scenarios in which the swarm steers into a certain direction other than fleeing from a predator were considered.

More recently, Hahn et al. (2019) investigated whether MARL can achieve similar results in a continuous environment without explicitly rewarding a certain distance to neighbors in *SELFish* (Swarm Emergent Learning Fish). Extending Morihiro et al. (2008), the predator is distracted by multiple agents in its vicinity as it randomly chooses its target from the agents nearby, enabling distraction as another survival strategy besides escaping. Small rewards are granted for surviving while getting caught by the predator is severely penalized. MARL policies were compared with behavior determined by the three Boids rules and the basic strategy of always moving in the opposite direction of the predator (*turn-away*).

This paper extends Hahn et al. (2019) and investigates whether swarm formations can be steered, for example to move into a certain direction or follow a target, solely using positive rewards in MARL. The biological equivalent to this task would be pursuit or foraging scenarios, in which multiple individuals pursuit a prey. While there is ample work on similar tasks in discrete domains such as grid worlds, e.g. Guo and Meng (2010), Zheng et al. (2017) and Baby et al. (2018), MARL is rarely used in continuous scenarios. And while Hüttenrauch et al. (2018) provide an in-depth investigation of end-to-end RL feature embeddings as state representations in a continuous multi-agent domain, they focus on improving the policies' performance but do not further analyze whether the emerging strategies combine successful problem solving with (natural) swarming.

Foraging Swarms

The goal of this paper is to investigate emergent behaviors in scenarios with multiple, selfishly acting agents. The learning problem is therefore defined in such a way that the agents have only indirect advantages by including the other agents in their behavior. Thus, the rewards received should not directly depend on the interaction with the other agents. Instead, the reward function is only used to give them the goal of pursuing a virtual point, hereinafter referred to as the target object, in the best possible way. The environmental conditions as well as the agents' possibilities for action and observation are described in more detail below. The most important data are summarized in the Table 1. The structure is strongly based on the scenario defined in SELFish; see Hahn et al. (2019).

Domain

The agents move in a square, two-dimensional world. The position of each agent is clearly defined by its x and y coordinates and its radius. These are real numbers, so the state space is continuous. At the beginning of each episode all agents and also the target object are initialized with random positions and orientations within the world. Afterwards they can rotate in any direction they choose at any time step. They move into that direction at constant speed. If agents or target objects leave the field borders, they reappear on the opposite side. The primary goal of the agents is to follow the target object as closely as possible. The target object moves randomly. Its direction of movement is changed in intervals of variable length in the range of 1 to 15 steps. The action that is executed after this time is also random and corresponds to the movement options that are also available to the agents, described in more detail in the Actions section. In order to simplify the task of the pursuing agents a bit, its movement speed is slightly lower. Since the target object is more a virtual point than a part of the physical world, collisions with it are not considered.

Table 1: The most important parameters of the domain.

Field size	60x60
Number of agents	25
Maximum perceived adjacent agents	8
Radius of perception	15
Radius of an agent	1
Movement distance of an agent per step	1.0
Movement distance of target object per step	0.95
Speed reduction in case of collision	50%

Actions

The action space A is discrete and comprises five different turning angles, between which the agents can choose as action: $\{-90°, -45°, 0°, +45°, +90°\}$. Within a time step, all agents choose one of the five actions and rotate in the corresponding direction. If they collide with another agent, i.e. the distance to the other agent is smaller than the added radii of both agents, they move forward at only half the speed. To make the collision handling more realistic, the agent is only slowed down if it is turned in the direction of the other at the time of the collision. This means that in situations where one agent hits another from behind, only the rear one of the two is slowed down. The agents cannot change their speed directly. However, they can indirectly slow down their speed in a certain direction by zigzagging, and thereby still have some influence on it.

Observations

Each agent receives information about the target object, itself, and the agents surrounding it. The data describes the distance and direction depending on the position and viewing direction of the observed entities and their global orientation, i.e. the direction in which they are currently moving. As mentioned in the Emergent Swarming section, swarms can form only with local information and can thereby be modeled as a partially observable learning problem. Thus, each observation $o \in O$ contains only part of the information of the overall state of the environment, which has been referred to as $s \in S$. This mapping from s to o is done by the observation model ξ. In the scenario defined here, this consists of limiting the perceptual range of an agent to its 8 nearest neighbors. The target object is invisible to the agent if it is not among the closest neighbors. In addition, an agent can only distinguish between distances that are below 30 units, i.e. half the width of the playing field. If another agent that is part of the observation is further away than this value, the distance to it in the observation is set to the maximum value of 30. All three pieces of information about distance, direction and angle of view are normalized and combined in a two-dimensional matrix. The first row is reserved for the target object, followed by the data about the own position, whereas actually only the current viewing direction is of interest. Distance and angle are therefore set to zero. The remaining rows describe the distance, direction and orientation of the other agents within the perceptual range. The architecture of the neural network requires that the observations have a fixed length. Therefore the lack of information about the position of the target object is modeled by replacing the corresponding part of the matrix with (-1) entries.

Reward

The reward that an agent e receives in each time step depends linearly on his distance to the target object. More precisely, it is calculated from the negative observed distance to the target object t, to which the maximum perceivable distance to the other agents or the target object is added. The result is then divided by the field width. The reward function is defined by:

$$r_e = \frac{-\textit{distance}(e,t) + \textit{max. obs. distance}}{\textit{field width}} \in [0, 0.5]$$

Other reward functions that we considered were to calculate the reward from the squared distance to the target object or to use the change of the absolute distance compared to the previous step. Giving the difference in distance as a reward initially accelerated the learning success, but the final learning results were slightly better for the approaches with absolute distance as a reward.

Training

The agents are trained using DQN. DQN (and its extensions) has the advantage that it is a versatile, well accepted

algorithm that is also provided in reinforcement learning libraries, like *TensorForce* (see Kuhnle et al. (2017)), which we used. Multi-agent reinforcement learning with single-agent algorithms is a recognized approach that can be used especially in application areas where guarantees of optimality are of secondary importance (see Matignon et al. (2012)). This is the case in the swarm scenario investigated here. Because of the homogeneity of the agents it seems reasonable that they also act according to the same strategy. In nature, the members of a swarm also follow behavior patterns that hardly differ from each other. This results in formations that appear to the outside world as a single entity.

Furthermore every agent has its own selfish objective of staying as close as possible to the virtual food source. That is why the learning procedure is such that only one agent learns at a time and its learned strategy is then used to control all agents. Thus, only one single neural network is trained, from which all agents derive their actions. This approach is called *Parameter Sharing*. The approach of letting only one agent of a homogeneous group learn was also used by Egorov (2016) and continued in *SELFish* (Hahn et al. (2019)), where it led to good learning outcomes.

See Table 2 for the set hyperparameters that achieved the most stable learning despite the non-static environment in which the agents are located.

Table 2: Selection of Hyperparameter for DQN.

Parameter	Value
Learning Steps	1200000
Steps per Episode	1000
Neural Network	
Hidden Layers	2
Neurons per Layer	64
Activation Function	ReLU
Learning Rate	0.003
Exploration	
Function	ϵ-Decay
Start Value	1.0
End Value	0.01
Half-lives	8
Discount Factor	0.999
Optimizer	Adam
Replay Buffer Capacity	100000
Batch Size	512

Simulations and Results

First of all, the behavioral patterns that have emerged through learning in the given scenario are discussed in the following. For an impression on the learned agent movement, please refer to Figure 1.[1] The position of the target object has a significant influence on the behavior of the agents. If it comes into the field of vision of an agent, the agent turns in its direction and moves after it. This means that the direction of movement of the agents in this area is the same and the density of the agents there is also significantly higher than in other areas of the environment. This alone is enough to create a swarm formation. It is noticeable that the agents maintain certain distances from each other, although collision are not prevented per se. They do slow down the speed of movement, however. In the event of a brief loss of visual contact with the target object, the agents orient themselves to the agents in their neighborhood. This is shown by the fact that the swarm around the target object always contains agents that do not currently have a view of it. The swarm cohesion therefore works quite well in most cases. However, there are always some agents that lose contact to the swarm. The behavior in this situation is inconsistent. Mostly they move straight ahead, but every now and then they adapt their movements to one of their neighbors. If the learned strategies are transferred to a scaled version of the scenario with more agents and a larger playing field, even more behaviors become visible. Here, the agents often form separate sub-flocks when separated from the main swarm.

The position of the neighboring agents also has some influence on the decisions of the individual. For example, in situations where there are too many agents around the target object, the agents behind often lose visual contact with the target. Nevertheless, they usually manage not to lose contact with the swarm over several steps.

Comparison with Static Rule-Based Agents

To better evaluate the learned behavior, it is helpful to compare the collected data with results of other strategies. For this purpose, the following three rule-based algorithms were implemented:

Simple If target object is visible, turn in the direction of the target, otherwise move straight ahead in the current viewing direction.

Boids Select an action that is calculated from a weighted combination of the principles of alignment, cohesion and separation with an additional force in the direction of the target object.

Random Perform a randomly selected action.

Since the Simple algorithm operates completely without information about the other agents in the environment, it serves as a reference value for the extent to which the agents adapt their behavior to the others. Although optimizations of this rule-based algorithm would still be possible, significant

[1] `https://youtu.be/eErmh_qFijw` shows the learned policies with and without collisions being penalized.

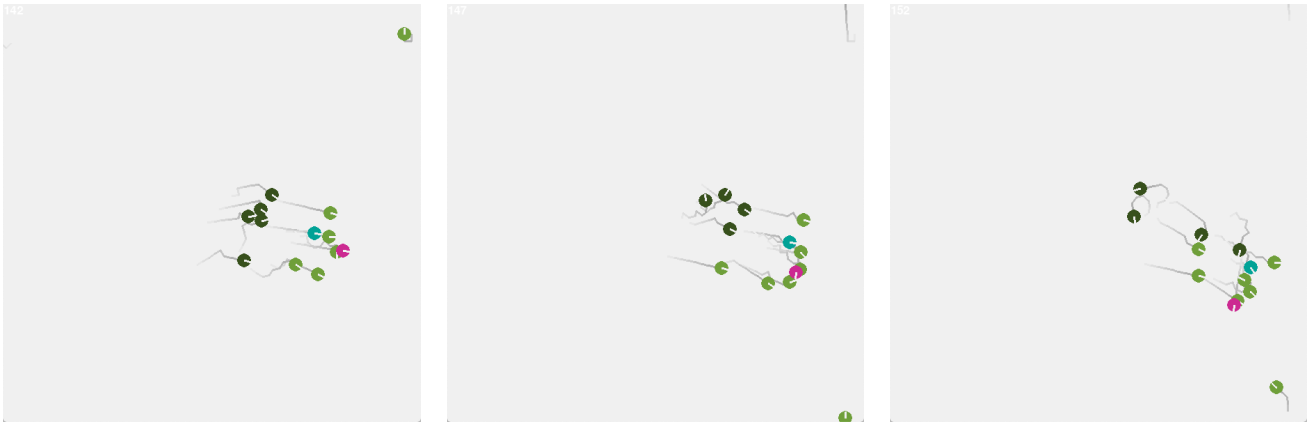


Figure 1: Typical behavior of agents without visual contact to the target object. The agents are represented by circles whose radius corresponds to the boundaries of their collision area. The direction of movement is indicated by a marker pointing from the center of the agent to the current viewing direction. The learning agent has a turquoise color, all other agents are colored green. The target object is visually distinguished from the agents by its pink color. If the agents have lost visual contact with the target object, they are colored darker. In addition, the agents have a tail, which indicates their movement in the last 10 steps. This tail is only used to better represent the time component in the still images and has no influence on the course of the experiment.

improvements over the Simple algorithm suggest that learning agents incorporate information about the observation of neighbors into their behavior.

The Boids algorithm serves as a recognized swarm simulation method for assessing how much the learned behavior resembles a natural swarm. In order to ensure comparability, this algorithm is also based only on the data that is also available to the learning agent. The force that directs the agent towards the target object is therefore only included in the calculation of the direction of movement if the target is among the agent's closest neighbors. This force is then combined, as described in the Emergent Swarming section, with the separation force, which ensures that the distances between the agents are maintained. The weighting of the repelling force depends quadratically on the distance to the neighbors. This results in very constant distances between the agents and there are few collisions.

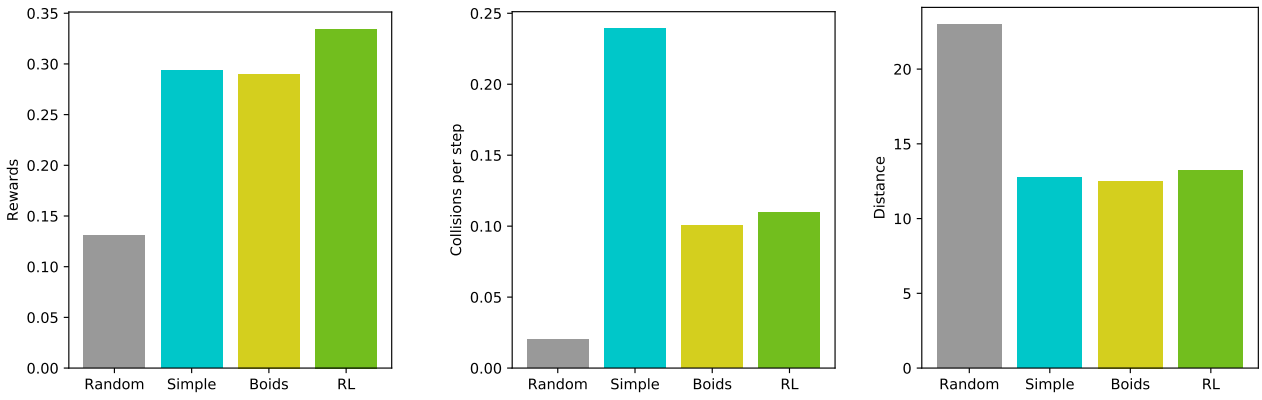
If there is no visual contact with the target, the rules of cohesion, alignment and separation are applied. Cohesion is weighted higher than the alignment of the viewing direction. The force of separation is quadratic as well, but overall it is stronger than before. This means that the distances between the agents are greater. This has a negative effect on swarm cohesion, but has the advantage that the target object is more likely to return to the area of sight of a neighboring agent after losing visual contact. The members of separated subflocks can thus more easily rejoin the main swarm.

Success in Pursuing the Target

In all of the algorithms studied, with the exception of the random algorithm, it is evident that the agents try to track the target object. They differ, however, in the way the agents behave in different situations and in the way the resulting swarm is structured. The swarm cohesion is most stable in the Boids algorithm. However, this does not necessarily mean that the tracking of the target object works particularly well here. To compare the different strategies in this aspect, one can look at the rewards that can be achieved by following the different strategies. The rewards that the agents receive depend linearly on their distance from the target. Therefore, high rewards can be used as a direct indicator of successful tracking of the target. The exact values are shown in Figure 2a. There you can see that all algorithms achieve significantly better results than random behavior. The highest values, however, are obtained by the behavior learned through reinforcement learning. This is followed by the other two strategies, which are about equally good despite their very different approaches.

Spacing and Collisions between Agents

One of the reasons for the Reinforcement Learner's higher rewards is that the agents have learned to avoid collisions with their neighbors. The frequency of collisions for the different algorithms is shown in Figure 2b. Collisions cause the agent to travel at half speed. Frequent collisions therefore make it much more difficult to track the target object, since the average speed of the agent is no longer sufficient to keep up. In the Simple strategy, the lack of coordination between the agents causes the agents to form a line behind the target object. Because they move a little faster than the target object, the agents often collide, which slows them down. Then they lose visual contact with the target. Most of the time



(a) Rewards (b) Collisions (c) Average distance between all agents

Figure 2: Comparison of the results of the different algorithms.

these situations lead to agents losing contact with the swarm completely. This is one of the main reasons why the Simple algorithm receives fewer reward overall than the Reinforcement Learner. Because of the repulsive force in the Boids algorithm the number of collisions is significantly lower for this strategy. The learned behavior has very similar values in this area. This shows that a form of collision avoidance has also been learned.

If we compare the average distance between all agents for the different action strategies (Figure 2c), they seem to have similar distances at first. However, a closer look at the frequency distribution of the distance to the nearest neighbor, which is shown in the Figure 3, reveals clear differences: The distances to the nearest agent, which remain very constant around the value 3 in the Boids algorithm, have a greater variance in the other two behaviors. Furthermore, it can be seen that the Simple algorithm also has some values below 2. Since the agents themselves have a radius of 1, collisions occur in steps where the distance to the nearest neighbor is less than 2. For agents trained by reinforcement learning, hardly any distances are recorded in the collision

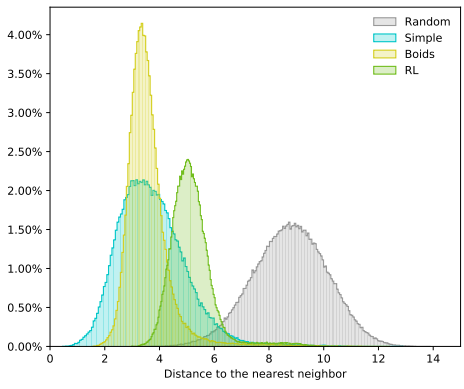


Figure 3: Distance to the nearest agent.

area. The observed behavior of the agents to avoid collisions is therefore also reflected in this diagram.

Loss of Visual Contact with the Target

Another possibility to use the information about the other agents is to deduce the position of the target object from their direction of movement. The agents also learn this, as outlined in Figure 1. The agents that have learned their behavior through reinforcement learning only lose visual contact with the target object at comparatively large distances and they also lose contact with the rest of the swarm less frequently. This is shown in the Figure 4. It depicts the distance of the agents in the steps after they have lost visual contact with the target. All distances were measured in the next 20 steps after the target object was out of the agent's range of perception, regardless of whether visual contact was restored in the following steps or not. With the learned behavior, the average distance to the target decreases again about 11 steps after having lost visual contact. This means that from this point on they usually get closer to the target object again. The agents of the other two algorithms, on the other hand, usually move further and further away from the target object as time progresses. With the Boids algorithm, the average distance from which they lose visual contact is still somewhat greater than with the Simple algorithm, but neither is able to approach the target object again and thus lose contact with the swarm.

With the Simple algorithm, the directions of movement and positions of the other agents are not included in the decisions. In this situation, the agent moves straight ahead. This is also one of the reasons why we only consider the next 20 steps, since the environment is finite, the agents might eventually run into the target again. However, since the target object has usually changed the direction of movement in the meantime, this does not result in visual contact being restored.

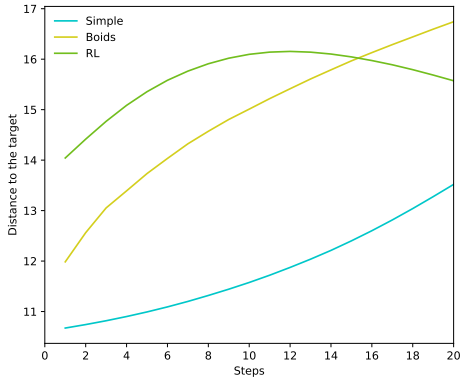


Figure 4: Development of the distance to the target object after loss of visual contact.

The opposite problem occurs with the Boids algorithm. Here the behavior is strongly adapted to that of the neighbors. Therefore, it almost never happens that individual agents are separated from the swarm. Instead, groups of several agents are always separated at the same time, which then stay together and follow the main swarm at some distance. The center of the neighboring agents, which determines the direction of movement in the Boids algorithm, always shifts a little towards the main swarm. Usually, however, this is not enough to bring the two groups together again. The parameter of the Boids algorithm were chosen such that the separation is weighted higher to increase the probability that the target object re-enters the observation radius after the contact is lost. However, the learned behavior of the Reinforcement Learner is more successful here. The Reinforcement Learner is not limited to a fixed weighting of the positions and viewing directions of the neighbors, but can adjust them according to the situation.

Influence of Neighbors

To further investigate the relationship between the agents, the scenario in which the agents learn was changed so that collisions between the agents are no longer handled. This means that they can overlap without reducing their movement speed. Agents that do not consider the positions of surrounding neighbors are not automatically disadvantaged. In this modified scenario it is therefore possible to directly compare strategies learned with and without information about the surrounding agents. In particular, the positive effects of considering neighbor positions can be shown. Two groups of agents were trained for the comparison, one of them lives in the original scenario, in which an agent has data on its 8 nearest neighbors. The other group, on the other hand, only has information about its own viewing direction and the position of the target object. Although these agents cannot see their neighbors, the target object is only visible to them if it is among the 8 nearest neighbors. This ensures that the two groups can be compared. For a better transferability of the hyperparameters of the neural network, the dimensions of the observation were not changed for this variation. The rows of the matrix, which were originally reserved for the positions of the other agents, are filled with (-1)-entries to represent a lack of information. This is thus handled in the same way as the situation where the target object is outside the observation range of the agent.

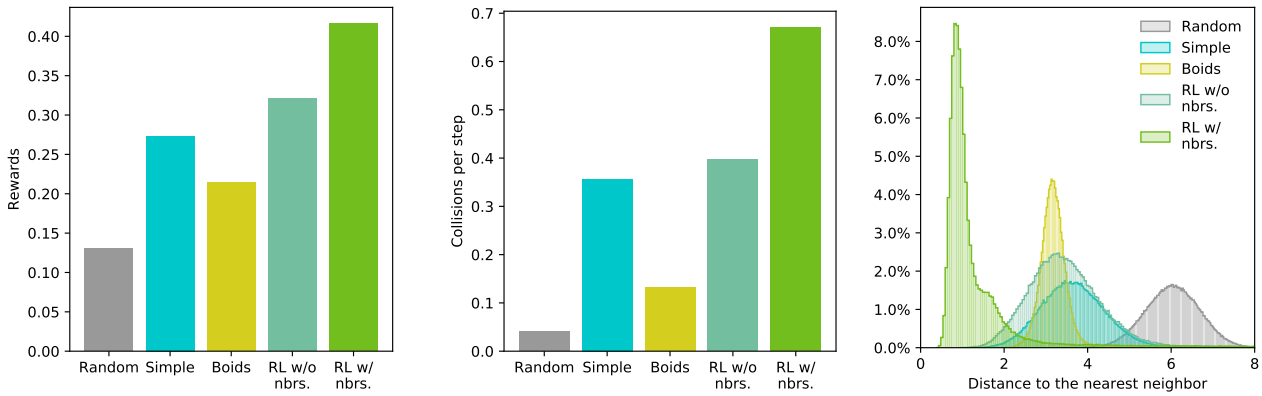
When comparing the rewards in Figure 5a, which are achieved by the agents with and without knowledge of their neighbors, it is clear that orientation to the neighbor has an advantage. The rewards of these agents are the highest. As expected, the number of potential collisions between the agents (which were counted but not penalized) is also much higher (Figure 5b), since they do not have any disadvantage from being very close to their neighbors. The distance to the nearest of the neighboring agents (Figure 5c) is also by far the lowest.

The behavior that the agents which cannot perceive their neighbors learn is essentially the same as the Simple algorithm. This means that the agents move behind the target object when they see it. In situations in which the target is outside their perceptual range, they move straight ahead. They also take into account the fact that the target object is moving, i.e. they head for a point that is slightly ahead of the current position of the target object. This means that they are even somewhat more successful in tracking the target object than the agents controlled by the Simple algorithm.

The collision avoidance explicitly integrated in the Boids algorithm has a negative effect on the success of this algorithm with respect to the rewards received. Accordingly, this strategy achieves the worst results with regard to the rewards after the random agent, as the Figure 5a shows.

Conclusion

We trained agents in a continuous, two-dimensional, partially observable environment on the objective to minimize their distance to a moving target using MARL. In comparison to previous work, which penalized being caught by a predator, we used positive reinforcement. Although every agent is self-interested in reaching its goal as there is no explicit incentive to collaborate, we showed that swarm behavior emerges among the agents. They have learned to orient themselves towards each other in situations when they lost sight of the target object in order to catch up, without being explicitly trained on this behavior. Furthermore, in a setting in which collisions are implicitly penalized by decelerating the colliding agents, they learned to keep a certain distance to each other. This behavior is analogous to the Boids rules observed by Reynolds in animals. That is why the learned behavior was compared with directly enforcing the Boids rules, which showed similar measurements in terms of collisions and distances between the agents. Despite the abstractness of the Boids rules, they lead to similar behavior as



(a) Rewards (b) Collisions (untreated) (c) Distance to nearest agent

Figure 5: Comparison of trained agents with and without knowledge of the movements of their neighbors with the rule-based algorithms in an environment without collision treatment.

our goal-oriented training. Thus, our work generates valuable insight in the emergence of swarming behavior without artificially enforcing it. Among that insight is a possible intention behind swarm forming: Our swarm behavior arises from a given set of goals for the agents. When they match Boids, we may argue that Boids swarm behavior as observed in nature might have evolved to achieve similar goals to the ones we gave our agents. Being able to generate this kind of swarming behavior out of pure self-interest could have applications in evacuation scenarios in which people collectively move to an emergency exit or in rescue scenarios, where multiple entities have to find and save victims.

References

Baby, S. A., Li, L., and Pokle, A. (2018). Analysis of emergent behavior in multi-agent environments. *CS234: Reinforcement Learning Winter 2018*.

Egorov, M. (2016). Multi-agent deep reinforcement learning. In *Stanford CS231n Course Report*.

Guo, H. and Meng, Y. (2010). Distributed reinforcement learning for coordinate multi-robot foraging. *J. Intell. Robotics Syst.*, 60(3-4):531–551.

Gupta, J., Egorov, M., and Kochenderfer, M. (2017). Cooperative multi-agent control using deep reinforcement learning. In *AAMAS Workshops*, pages 66–83.

Hahn, C., Phan, T., Gabor, T., Belzner, L., and Linnhoff-Popien, C. (2019). Emergent escape-based flocking behavior using multi-agent reinforcement learning. In *Conference on Artificial Life (ALIFE 2019)*.

Hüttenrauch, M., Sosic, A., and Neumann, G. (2018). Deep reinforcement learning for swarm systems. *CoRR*, abs/1807.06613.

Kuhnle, A., Schaarschmidt, M., and Fricke, K. (2017). Tensorforce: a tensorflow library for applied reinforcement learning. Web page.

Leibo, J. Z., Zambaldi, V., Lanctot, M., Marecki, J., and Graepel, T. (2017). Multi-Agent Reinforcement Learning in Sequential Social Dilemmas. In *Proceedings of the 16th Conference on Autonomous Agents and Multiagent Systems*, pages 464–473. IFAAMAS.

Matignon, L., Laurent, G. j., and Le fort piat, N. (2012). Review: Independent reinforcement learners in cooperative markov games: A survey regarding coordination problems. *Knowl. Eng. Rev.*, 27(1):1–31.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., and Hassabis, D. (2015). Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533.

Morihiro, K., Nishimura, H., Isokawa, T., and Matsui, N. (2008). Learning grouping and anti-predator behaviors for multi-agent systems. In *Knowledge-Based Intelligent Information and Engineering Systems*, pages 426–433. Springer Berlin Heidelberg.

O'Grady, R., Pinciroli, C., Groß, R., Christensen, A. L., Mondada, F., Bonani, M., and Dorigo, M. (2009). Swarm-bots to the rescue. In *European Conference on Artificial Life*, pages 165–172. Springer.

Reynolds, C. W. (1987). Flocks, herds and schools: A distributed behavioral model. In *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '87, pages 25–34, New York, NY, USA. ACM.

Reynolds, C. W. (1999). Steering behaviors for autonomous characters. `https://www.red3d.com/cwr/steer/gdc99/`.

Watkins, C. J. and Dayan, P. (1992). Q-Learning. *Machine learning*.

Zheng, L., Yang, J., Cai, H., Zhang, W., Wang, J., and Yu, Y. (2017). Magent: A many-agent reinforcement learning platform for artificial collective intelligence. *CoRR*, abs/1712.00600.