Multiagent Leader Based Learning for Complex Swarm Coordination

Everardo Gonzalez
gonzaeve@oregonstate.edu
Oregon State University
Corvallis, Oregon, USA

Andrew Festa
festaa@oregonstate.edu
Oregon State University
Corvallis, Oregon, USA

Kagan Tumer
ktumer@oregonstate.edu
Oregon State University
Corvallis, Oregon, USA

ABSTRACT

Intelligent swarms offer potential solutions to real world problems that require large numbers of coordinated agents acting in unison in hostile environments. However, learning those coordinated actions is challenging in a swarm because each agents' exploratory action acts as "noise" to other agents in the system. Fitness shaping techniques aim to reduce this noise by generating agent specific fitnesses, while swarm shepherding techniques aim to reduce this noise by limiting the number of learning agents. However, neither addresses how to coordinate a swarm to achieve multiple tasks simultaneously. In this work, we introduce Multiagent Leader Based Learning (MALBL) where we investigate how a subset of learning agents can manipulate and split the swarm to address multiple tasks. The key insight of this work is to shape the fitness of a leader not only by its own impact, but also those of its followers. We demonstrate that MALBL outperforms state of the art multiagent learning techniques on a complex swarm objective that requires tight coordination on multiple tasks.

CCS CONCEPTS

• **Computing methodologies → Multi-agent systems**; **Cooperation and coordination**; *Evolutionary robotics.*

KEYWORDS

multiagent, swarm, coevolution, fitness shaping, tight coupling

ACM Reference Format:
Everardo Gonzalez, Andrew Festa, and Kagan Tumer. 2018. Multiagent Leader Based Learning for Complex Swarm Coordination. In *Woodstock '18: ACM Symposium on Neural Gaze Detection, June 03–05, 2018, Woodstock, NY*. ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

Intelligent swarm systems offer interesting solutions for domains such as underwater exploration, oil spill cleanup, and disaster recovery, where large numbers of agents must coordinate in an environment hostile to humans. However, the interactions between many learning agents in a swarm create a noisy feedback signal that is particularly difficult to learn from in environments with sparse and uninformative feedback.

The noise in this system feedback makes it unreliable as a learning signal for individual agents. This signal does not give any insight into whether or not a particular agent is making a useful contribution to the system. For instance, an underwater robot might explore an area that a teammate already explored, and this robot would still be rewarded despite it contributing nothing new to the system's objective. Ideally, we would want this agent's feedback to push it to explore a different area that has not been covered by any other agents.

Techniques focused on shaping fitness functions in multiagent systems partially address this problem for multiagent learning by distilling the system feedback into cleaner learning signals for individual agents that push each agent to maximize its own individual contribution to the system. Swarm shepherding techniques take a different approach by deploying a single learning agent to learn a leader policy for a subset of swarm leaders that can guide a much larger set of swarm followers with preset policies, effectively sidestepping the problem of agents disrupting each other's learning signals.

A key limitation of these techniques is that neither can coordinate a swarm to achieve a complex objective that requires tight coordination on multiple tasks. A purely multiagent learning approach, even with shaped fitness functions, fails to scale up to the large number of agents in a swarm. On the other hand, a swarm shepherding approach only works to guide the entire swarm to achieve one task, and does not have any way of splitting up the swarm to tackle multiple tasks.

In this work, we introduce Multiagent Leader Based Learning (MALBL) where agents are split into swarm followers with preset policies and swarm leaders that individually learn to guide followers to accomplish a complex system objective. We take a multiagent learning approach to swarm shepherding where each leader receives a shaped fitness based on how that particular leader influences followers in the swarm, which makes it possible for leaders to specialize in different aspects of the swarm's objective.

The key insight is that we can actually achieve complex coordination for an entire swarm with only a small subset of the swarm actually learning control policies. We can keep the number of learners in the system low without compromising the capability of the swarm.

The primary contributions of this paper are:

(1) Multiagent system formulation of leader-follower swarms where leaders each learn a unique policy to tackle different swarm tasks.

Woodstock '18, June 03–05, 2018, Woodstock, NY
© 2018 Association for Computing Machinery.
ACM ISBN 978-1-4503-XXXX-X/18/06...$15.00
https://doi.org/10.1145/1122445.1122456

(2) Fitness shaping technique that accounts for how leaders impact their followers so leaders can contribute to better overall swarm performance.

Multiagent leader based learning outperforms standard multiagent learning with state of the art reward shaping techniques on a complex objective that requires tight coordination amongst many agents.

2 BACKGROUND

2.1 Cooperative Co-Evolutionary Algorithms

Cooperative Coevolutionary Algorithms (CCEAs) make it possible to apply an evolutionary approach to learning joint-policies for a multiagent system. Whereas an Evolutionary Algorithm (EA) evolves a population of policies for a single agent, a CCEA coevolves several subpopulations of policies, one subpopulation for each agent in the multiagent system. This essentially splits the learning problem into several subproblems, where learning each agent's policy is a specific subproblem. A primary consequence of splitting up the learning problem this way is that evaluating fitnesses becomes more complex, as a policy can only be evaluated as part of a joint-policy, not on its own. As such, a policy is always evaluated in the context of how it performs with policies from other subpopulations.

A CCEA initializes one sub-population of k neural networks for each agent operating in the system. Similar to a standard EA, at each generation, the CCEA generates k successor networks by mutating the weights contained in each subpopulation, bringing the subpopulation size to *2k*. For evaluating the overall fitness of the system, neural networks are selected without replacement and placed on teams. Each team is evaluated according to the system evaluation function to calculate the fitness of that team. Each network on that team is assigned a score based on this fitness value as well as some shaping function specific to that individual. From there, each network has an assigned fitness that each EA can downselect from to bring the subpopulation back down to k networks and prepare for the next generation.

2.2 Shaping Multiagent Fitness Functions

One of the primary challenges with applying cooperative co-evolutionary algorithms to multiagent learning is in distilling the system feedback signal into individual agent-specific feedback signal. Reward shaping methods from Multiagent Reinforcement Learning are directly applicable to these problems in the form of a shaped fitness function [1, 3, 4, 10]. We demonstrate this mathematically in Equation 1.

$$f_i = G(z) + F(z)_i \tag{1}$$

In the above equation, $G(z)$ represents the fitness from the system objective function, and $F(z)_i$ represents additional information added by the fitness shaping technique for agent i's fitness. f_i is the shaped fitness function that would be given to individual agent i. Difference objectives are one method of injecting information via shaping this fitness function. A difference objective compares the actual outcome of a team against a counterfactual outcome that did not include a particular agent's trajectory in order to determine that agent's portion of the contribution to the system.

Different objectives are calculated according to equation 2. The actual fitness of the team is calculated as $G(z)$. A counterfactual outcome is created by replacing agent i's trajectory within the team with a counterfactual trajectory for this agent, c_i. This counterfactual outcome is represented by $G(z_{-i} \cup c_i)$. By subtracting the fitness of the counterfactual team from the fitness of the actual team, we can calculate the impact that agent i had on the system.

$$D_i = G(z) - G(z_{-i} \cup c_i) \tag{2}$$

In some multiagent domains, tasks can be tightly coupled, meaning that several agents must stumble upon the correct sequence of joint-actions necessary to accomplsh the task before the system receives any positve feedback for working towards that task. D_{++} introduces the concept of "stepping stone" feedback in order to extend difference evaluation functions to these problems. Rather than subtracting an agent from the system to determine its contribution to the system, D_{++} introduces multiple counterfactual partners for an agent to determine what that agent could have contributed had the agent had help with the tasks it attempted to complete during training. The D_{++} fitness function is calculated according to equation 3.

$$D^n_{++}(i) = \frac{G(z_{+(\cup_{i=1,\dots,n})i}) - G(z)}{n} \tag{3}$$

In the above equation, $G(z_{+(\cup_{i=1,\dots,n})i}$ represents the counterfactual team fitness with n partner agents added to the system, $G(z)$ represents the true fitness of the team, and n is used to discount the fitness according to the number of added partner agents.

While these techniques for shaping agent fitness functions have shown great success on multiagent teams, they fail to scale up when a team has many, many agents which have to perform many tightly coupled tasks. D_{++} becomes computationally expensive to compute with many counterfactual parter agents, and does not scale effectively with really tight coupling requirements. Standard difference objective functions on their own fail to extend to tightly coupled domains in the first place.

2.3 Swarm Shepherding

Swarm shepherding techniques focus on guiding a swarm to a particular point in space. Many approaches formulate a swarm as being composed of shepherds and sheep [8], which in our work we consider as leaders and followers, respectively. Followers are simple agents that exhibit simple prescribed swarm behaviors based on local interactions with other swarm members. A leader uses either a prescribed control policy [5–7], or a learned control policy to guide the followers to a predefined point [2, 11, 12]. When the leaders' control policy is learned, one policy is learned that is copied to all of the leaders.

While recent work has addressed many challenges to make learning more robust for these guidance problems, there has been little investigation into how to learn to shepherd a swarm when the swarm must split up to cover multiple points simultaneously. Such coordination would require a multiagent learning approach where each leader would learn a unique control policy to split off and guide a subset of the swarm to one of the points of interest.

Nguyen et al. begin to explore the potential benefits of combining multiagent learning with swarm shepherding by having each leader learn its own control policy to guide a swarm, and demonstrate that this specialization of each leader results in more effective swarm coordination [9]. However, their approach is limited to guiding the entire swarm to one location, and does not consider how this specialization could be leveraged to split the swarm amongst several points of interest.

3 MULTIAGENT LEADER BASED LEARNING

Tasks requiring tightly coupled coordinated behaviors between many agents is difficult for agents to learn due to positive feedback only occurring when multiple agents simultaneously act in a coordinated manner. From the perspective of a learning agent, this would require multiple random sequences of actions to happen to achieve the task, or some part of the task, that is enough to provide a positive feedback signal from the environment. Particularly as the number of required agents grows, this becomes exponentially more unlikely with every added agent.

Consider a task where multiple agents must simultaneously observe a point of interest (POI) in order to receive any reward from the environment.There are multiple POIs scattered throughout the environment, some further awy than others. If a single POI requires three agents to observe it, then three agents would have to pick actions such that they are within the observation radius of the POI at the same time. The further the POI, the less likely a sequence of random actions from multiple agents, each with a different starting location, will bring them to a similar location. For two or three agents, this is unlikely. As this coupling requirement increases, this random coordination to receive any initial positive feedback becomes next to impossible.

This work introduce multiagent leader-based learning as a method for addressing this necessity of agents having to randomly discover a set of coordinated behaviors for tightly coupled tasks requiring many agents. The method splits agents into two types: leaders and followers. Leaders take on the form of typical learning agents that take the state as input and produce an action as output at every time step. Followers have the same state and action spaces as the leaders, but instead they use a simple preset policy that causes them to move towards nearby agents while maintaining a minimal distance between each other.

The key insight here is that the follower policy acts as a method of injecting domain knowledge about the task without fully specifying the behavior of the system. In a tightly coupled problem, multiple agents must work in close coordination to accomplish the task. The follower policy pushes some agents towards acting in a manner that is conducive to the agents working closely. Often, designers will shape the fitness functions to try and capture how well a task is performed, and it is this fitness shaping that is meant to drive the manifestation of a desired behavior. However, simple policies themselves can also serve as an effective means of guiding systems of agents to coordinate in complex manners.

While the leader-follower paradigm is able to guide agents to establishing and maintaining coordination in complex tasks, there remains a problem of effectively assigning credit to the leaders so that they can learn to optimize the solution. The problem of credit assignment exists in any multiagent learning problem, especially in the case of episodic rewards. In these cases, it can be extremely difficult to tease out an individual agent's contribution to the task. With the leader-follower paradigm, this problem can become exacerbated as a leader's actions are also responsible for the actions of nearby followers.

We extend the idea of difference rewards to capture this larger impact leaders have on the system.

4 EXPERIMENTS

5 RESULTS

6 CONCLUSION

ACKNOWLEDGMENTS

The grant that funded this work.

REFERENCES

[1] Jacopo Castellini, Sam Devlin, Frans A Oliehoek, and Rahul Savani. 2022. Difference rewards policy gradients. *Neural Computing and Applications* (2022), 1–24.

[2] Essam Debie, Hemant Singh, Saber Elsayed, Anthony Perry, Robert Hunjet, and Hussein Abbass. 2021. A Neuro-Evolution Approach to Shepherding Swarm Guidance in the Face of Uncertainty. In *2021 IEEE International Conference on Systems, Man, and Cybernetics (SMC).* 2634–2641. https://doi.org/10.1109/SMC52423.2021.9659082

[3] Sam Devlin and Daniel Kudenko. 2011. Theoretical considerations of potential-based reward shaping for multi-agent systems. In *The 10th international conference on autonomous agents and multiagent systems.* ACM, 225–232.

[4] Sam Michael Devlin and Daniel Kudenko. 2012. Dynamic potential-based reward shaping. In *Proceedings of the 11th international conference on autonomous agents and multiagent systems.* IFAAMAS, 433–440.

[5] Kaoru Fujioka and Sakiko Hayashi. 2016. Effective shepherding behaviours using multi-agent systems. In *2016 IEEE Region 10 Conference (TENCON).* 3179–3182. https://doi.org/10.1109/TENCON.2016.7848636

[6] Jyh-Ming Lien, O.B. Bayazit, R.T. Sowell, S. Rodriguez, and N.M. Amato. 2004. Shepherding behaviors. In *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04. 2004*, Vol. 4. 4159–4164 Vol.4. https://doi.org/10.1109/ROBOT.2004.1308924

[7] Jyh-Ming Lien, S. Rodriguez, J. Malric, and N.M. Amato. 2005. Shepherding Behaviors with Multiple Shepherds. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation.* 3402–3407. https://doi.org/10.1109/ROBOT.2005.1570636

[8] Nathan K. Long, Karl Sammut, Daniel Sgarioto, Matthew Garratt, and Hussein A. Abbass. 2020. A Comprehensive Review of Shepherding as a Bio-Inspired Swarm-Robotics Guidance Approach. *IEEE Transactions on Emerging Topics in Computational Intelligence* 4, 4 (2020), 523–537. https://doi.org/10.1109/TETCI.2020.2992778

[9] Tung Nguyen, Jing Liu, Hung Nguyen, Kathryn Kasmarik, Sreenatha Anavatti, Matthew Garratt, and Hussein Abbass. 2020. Perceptron-Learning for Scalable and Transparent Dynamic Formation in Swarm-on-Swarm Shepherding. In *2020 International Joint Conference on Neural Networks (IJCNN).* 1–8. https://doi.org/10.1109/IJCNN48605.2020.9207539

[10] Aida Rahmattalabi, Jen Jen Chung, Mitchell Colby, and Kagan Tumer. 2016. D++: Structural credit assignment in tightly coupled multiagent domains. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS).* 4424–4429. https://doi.org/10.1109/IROS.2016.7759651

[11] Ovunc Tuzel, Gilberto Antonio Marcon dos Santos, Chloë Fleming, and Julie Adams. 2018. *Learning Based Leadership in Swarm Navigation: 11th International Conference, ANTS 2018, Rome, Italy, October 29–31, 2018, Proceedings.* 385–394. https://doi.org/10.1007/978-3-030-00533-7_33

[12] Jixuan Zhi and Jyh-Ming Lien. 2021. Learning to Herd Agents Amongst Obstacles: Training Robust Shepherding Behaviors Using Deep Reinforcement Learning. *IEEE Robotics and Automation Letters* 6, 2 (2021), 4163–4168. https://doi.org/10.1109/LRA.2021.3068955