Indirect Credit Assignment in a Multiagent System

Extended Abstract

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

Siddarth Viswanathan
Cal Poly State University
San Luis Obispo, United States
sviswa01@calpoly.edu

Kagan Tumer
Oregon State University
Corvallis, United States
kagan.tumer@oregonstate.edu

ABSTRACT

Learning in a multiagent system requires structural credit assignment to distill system performance into agent-specific feedback. Fitness shaping methods largely isolate agent credit, but struggle when an agent's actions do not directly affect system feedback. This work introduces D-Indirect, a fitness shaping method that gives credit for both direct actions and actions that have an indirect impact on the system's performance. We demonstrate the effectiveness of D-Indirect in a simulated shepherding scenario and our results show that learning with D-Indirect significantly outperforms learning with the standard difference evaluation and the system evaluation when agents indirectly impact system performance.

KEYWORDS

Fitness Shaping; Reward Shaping; Swarm Shepherding

ACM Reference Format:
Everardo Gonzalez, Siddarth Viswanathan, and Kagan Tumer. 2024. Indirect Credit Assignment in a Multiagent System: Extended Abstract. In *Proc. of the 23rd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2024), Auckland, New Zealand, May 6 – 10, 2024*, IFAAMAS, 3 pages.

1 INTRODUCTION

Multiagent systems require agents to cooperate in order to maximize performance from a system evaluation function. These systems are well suited for scenarios such as search and rescue [8, 12], oil spill cleanup [9, 14], and traffic congestion [13]. Learning to coordinate in a system like this is difficult because the system evaluation captures the performance of the system overall, not of individual agents. This creates the problem of structural credit assignment, which is ensuring that each agent receives feedback based on its individual actions [1, 10].

Fitness shaping methods help remedy the structural credit assignment problem. The difference evaluation in particular shapes the system evaluation for an agent based on how that agent's actions impacted the system evaluation function [1, 4]. This largely isolates credit for individual agents. However, the difference evaluation cannot assign credit for actions that indirectly map to the system evaluation. For instance, if an agent only takes actions to enable *other* agents to increase system performance, then the difference evaluation would provide no feedback for that agent.

Proc. of the 23rd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2024), N. Alechina, V. Dignum, M. Dastani, J.S. Sichman (eds.), May 6 – 10, 2024, Auckland, New Zealand. © 2024 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org).

This work introduces a new fitness shaping method: "D-Indirect". D-Indirect uses the same principle as the difference evaluation, but also accounts for indirect credit assignment. This gives an agent feedback based on both its own actions *as well as* the actions of other agents that were influenced by that agent. The key insight is that even if an agent's actions do not directly map to the system evaluation, these actions could still impact the system evaluation downstream. We can capture this indirect mapping by incorporating agent-to-agent interactions in order to make sure these actions are appropriately rewarded. The key contribution of this work is D-Indirect, a fitness shaping method that provides an agent with an individual feedback signal for its overall contribution to a system including its influence on other agents.

2 BACKGROUND

The difference evaluation is a fitness shaping method that computes direct credit assignment in a multiagent system and has successfully improved performance in various domains [1–3]. By comparing the system evaluation to a counterfactual evaluation with agent i's actions removed, we can isolate agent i's direct impact on system performance. This is shown below.

$$D_i = G(z) - G(z_{-i} \cup c_i) \tag{1}$$

where D_i is the difference evaluation for agent i, $G(z)$ is the system evaluation with all agents' actions, and $G(z_{-i} \cup c_i)$ is the counterfactual evaluation with agent i's actions replaced with null actions c_i. The structure of D is powerful because it gives an agent feedback that is both sensitive to that agent's actions and aligned with G. The issue with D is that it requires a direct mapping from agent i's actions to the output of G. If an agent's actions do not directly affect the calculation of the system evaluation, then D provides no useful feedback.

3 D-INDIRECT

D-Indirect borrows the structure of D, and modifies it in order to compute indirect credit assignment in a multiagent system. Rather than simply removing agent i in the counterfactual evaluation, D-Indirect removes agent i as well as other agents that were *influenced* by agent i. The motivating idea is that even if an agent's actions do not directly impact the system evaluation, that agent can still have important interactions with other agents that we can measure as "influence". This makes it so that if agent i takes actions that enable agent i''s actions to have a direct impact, then agents i and i' both get credit for those actions.

We represent D-Indirect as $D^{Indirect}$, and compute it by taking Equation 1 from the standard difference evaluation and modifying the counterfactual system evaluation. Rather than replacing only

agent i with a counterfactual, agents in the set F_i are replaced with a counterfactual. F_i is the set of agent i and agents influenced by agent i. We compute $D^{Indirect}$ according to the following equation.

$$D_i^{Indirect} = G(z) - G(z_{-F_i} \cup c_{F_i}) \tag{2}$$

In this work, we use a null counterfactual for c_{F_i}, so actions from agents F_i are simply removed. F_i is determined by an influence function that computes which agents were influenced by agent i, and must be defined according to how agents interact in the domain.

4 EXPERIMENTAL SETUP

We set up our experiment in the multiagent multi-POI shepherding domain. Shepherd agents must learn to guide preprogrammed (**non-learning**) sheep agents to various points of interest (POIs) in order to maximize G. The challenge is that only sheep agents' actions directly impact G, so shepherd agents must learn despite having no direct impact on system performance. The sheep behave according to Reynold's flocking behaviors [11]. The computation of G is shown below.

$$G(s_{final}) = \frac{1}{p} \sum_j \frac{1}{\min_i(dist(sheep_i, POI_j))} \tag{3}$$

G is evaluated based on the final state of the system after one episode containing 200 timesteps. s_{final} includes the final positions of all shepherd and sheep agents in the system, and p is the number of POIs. The summation is across all POIs and sheep i is the closest sheep agent to POI j. In order to replace the actions of an agent i or set of agents F_i with a null counterfactual to compute $G(z_i \cup c_i)$ or $G(z_{-F_i} \cup c_{F_i})$, respectively, we remove the final state of those agents from the computation of G shown above. This is equivalent to replacing the agents' actions because agents begin in the same initial state at the beginning of every episode.

The map has an (X,Y) size of (110, 100). Shepherd and sheep agents are placed in 10 pairs near the bottom Y edge of the map. 10 POIs are scattered according to a uniform distribution from (10,30) to (100, 90). Each shepherd must move a short distance to begin influencing its nearest sheep agent, and then must maintain influence over that sheep agent to bring it towards a POI. To achieve the maximum score of $G = 1$, each shepherd would have to bring one sheep to one POI. The set F_i for shepherd agent i includes that shepherd agent as well as the sheep agents that remained closest to that particular shepherd.

The shepherd policies are learned using a Cooperative Coevolutionary Algorithm (CCEA) [5–7]. Each shepherd agent evolves a population of 50 neural networks with 2 layers and 9 units per layer. For evaluation, each shepherd policy is placed on a randomly formed team without replacement, and scored according to the fitness shaping method: G (no shaping), D, or $D^{Indirect}$. Selection is done according to n-elites with binary tournament selection. The top 5 policies of a shepherd's population are added to the new population, and the higher performing of two randomly chosen policies is mutated and added until the new population is filled. An additional team is formed using the highest performing policy of each population to evaluate the system performance.

Each trial of the CCEA was run for 500 generations. Each shaping method (G, D, and $D^{Indirect}$) was used for 20 independent trials.

5 RESULTS AND DISCUSSION

Figure 1 shows the average performance and standard error for each shaping method. Symbols are included to represent average performance every 100 generations to help provide clarity. $D^{Indirect}$ results in higher performing joint policies for the shepherd agents than G or D. $D^{Indirect}$ outperforms G because it gives each shepherd agent individualized feedback based on how its actions influenced sheep agents to move towards POIs. On the other hand, G provides each shepherd agent with the entire system feedback, so there is no distinction between which agents contributed to the system and which did not. Instead, each shepherd agent is evaluated based on how its randomly assigned team performed rather than how that specific agent performed. The shepherd agents can still learn with G, but not as effectively as with $D^{Indirect}$. Learning with D results in consistently useless joint policies because there is no direct mapping from shepherd agents' actions to G. This means that D consistently evaluates to 0 for each shepherd's policy because there is no difference in the computation of G when a shepherd agent's actions are removed.

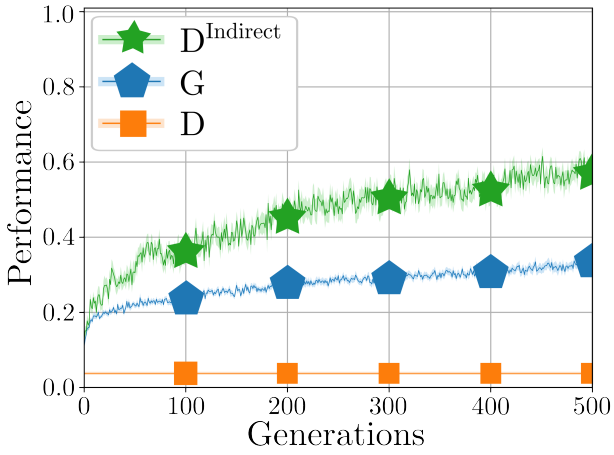

Figure 1: Learning curves with different fitness shaping methods. We see that $D^{Indirect}$ learns much faster than G, and D results in no learning. This is because $D^{Indirect}$ gives shepherd agents feedback based on their indirect impacts to system performance whereas G provides no shaping, and D provides no feedback for shepherd agents.

D-Indirect makes it possible to address structural credit assignment for indirect actions. By accounting for how an agent's actions influence other agents, D-Indirect gives that agent credit for those influencing actions that would otherwise be ignored in shaping. Future work should investigate further how agent-to-agent influence can be leveraged for more effective fitness shaping.

ACKNOWLEDGMENTS

This work was partially supported by the National Science Foundation with grant No. CNS-1950927 and the Air Force Office of Scientific Research with grant No. FA9550-19-1-0195.

REFERENCES

[1] Adrian Agogino and Kagan Tumer. 2004. Efficient Evaluation Functions for Multi-rover Systems. In *Genetic and Evolutionary Computation – GECCO 2004*, Kalyanmoy Deb (Ed.). Springer, Berlin, Heidelberg, 1–11.

[2] Adrian K. Agogino and Kagan Tumer. 2008. Analyzing and visualizing multiagent rewards in dynamic and stochastic domains. *Autonomous Agents and Multi-Agent Systems* 17, 2 (Oct. 2008), 320–338. https://doi.org/10.1007/s10458-008-9046-9

[3] Adrian K Agogino and Kagan Tumer. 2012. A multiagent approach to managing air traffic flow. *Autonomous Agents and Multi-Agent Systems* 24 (2012), 1–25.

[4] Mitchell K Colby and Kagan Tumer. 2012. Shaping fitness functions for coevolving cooperative multiagent systems. *Autonomous Agents and Multi-Agent Systems* 1 (2012), 425–432.

[5] Joshua Cook and Kagan Tumer. 2021. Ad hoc teaming through evolution. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion* (Lille, France) *(GECCO '21)*. Association for Computing Machinery, New York, NY, USA, 89–90. https://doi.org/10.1145/3449726.3459560

[6] Joshua Cook and Kagan Tumer. 2022. Fitness shaping for multiple teams. In *Proceedings of the Genetic and Evolutionary Computation Conference* (Boston, Massachusetts) *(GECCO '22)*. Association for Computing Machinery, New York, NY, USA, 332–340. https://doi.org/10.1145/3512290.3528829

[7] Joshua Cook, Kagan Tumer, and Tristan Scheiner. 2023. Leveraging Fitness Critics To Learn Robust Teamwork. In *Proceedings of the Genetic and Evolutionary Computation Conference* (Lisbon, Portugal) *(GECCO '23)*. Association for Computing Machinery, New York, NY, USA, 429–437. https://doi.org/10.1145/3583131.3590497

[8] Daniel S. Drew. 2021. Multi-Agent Systems for Search and Rescue Applications. *Current Robotics Reports* 2, 2 (June 2021), 189–200. https://doi.org/10.1007/s43154-021-00048-3

[9] Ellips Masehian and Mitra Royan. 2015. Cooperative Control of a Multi Robot Flocking System for Simultaneous Object Collection and Shepherding. In *Computational Intelligence*, Kurosh Madani, António Dourado Correia, Agostinho Rosa, and Joaquim Filipe (Eds.). Springer International Publishing, Cham, 97–114.

[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)*. IEEE, 4424–4429. https://doi.org/10.1109/IROS.2016.7759651

[11] Craig W. Reynolds. 1987. Flocks, herds and schools: A distributed behavioral model. In *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '87)*. Association for Computing Machinery, New York, NY, USA, 25–34. https://doi.org/10.1145/37401.37406

[12] Samy A. Shedied. 2013. Optimal trajectory planning for the herding problem: a continuous time model. *International Journal of Machine Learning and Cybernetics* 4, 1 (Feb. 2013), 25–30. https://doi.org/10.1007/s13042-012-0071-2

[13] Kagan Tumer, Zachary T Welch, and Adrian Agogino. 2008. Aligning social welfare and agent preferences to alleviate traffic congestion. In *Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems - Volume 2* (Estoril, Portugal) *(AAMAS '08)*. International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 655–662.

[14] Xudong Ye, Bing Chen, Pu Li, Liang Jing, and Ganning Zeng. 2019. A simulation-based multi-agent particle swarm optimization approach for supporting dynamic decision making in marine oil spill responses. *Ocean & Coastal Management* 172 (2019), 128–136. https://doi.org/10.1016/j.ocecoaman.2019.02.003