The Specificity vs. Expense Trade-Off of Multiagent Credit

Raghav Thakar
thakarr@oregonstate.edu
Collaborative Robotics and Intelligent Systems Institute
Oregon State University
Corvallis, Oregon, USA

Kagan Tumer
kagan.tumer@oregonstate.edu
Collaborative Robotics and Intelligent Systems Institute
Oregon State University
Corvallis, Oregon, USA

Abstract

Many real-world settings call for multiple autonomous agents to coordinate and work as a collective. In such settings, cooperative coevolution—which trains a policy for each agent in parallel—has emerged as a popular choice. However, providing these coevolving agents only with feedback that evaluates the whole system of agents is often suboptimal, as learning is each individual agent's individual responsibility. This necessitates deriving agent-specific feedback from the system feedback, termed the "multiagent credit assignment problem". Oftentimes, however, such isolation of agent-specific feedback may be computationally expensive, and may not yield commensurate improvements in learning efficiency. In this paper, we present D^{flex}, an extension of Difference Evaluation that offers tuneable specificity in the feedback computed for each agent. This tuneability—achieved by allowing a subset of coevolving agents to 'share' feedback—allows practitioners to trade between feedback specificity and its compute expense. Our initial results in the multiagent rover exploration problem not only empirically indicate the presence of the feedback specificity vs. compute cost trade-off, but also show D^{flex}'s ability to provide control over it.

CCS Concepts

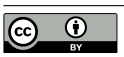
• **Computing methodologies** → **Cooperation and coordination**; **Multi-agent systems**; *Intelligent agents.*

Keywords

Multiagent Learning, Multiagent Credit Assignment, Cooperative Coevolutionary Algorithms

1 Introduction

Many real-world settings are best modelled as a multiagent system, where multiple agents simultaneously learn to coordinate and perform complex tasks. Cooperative coevolution, wherein a policy for each agent in the system is learnt in parallel, has emerged as a popular learning paradigm in such settings. Oftentimes, however, the only available feedback to these coevolving agents evaluates the performance of the whole system, and not each agent individually. For agents to learn efficiently, it is necessary to provide feedback that evaluates each agent individually. Without this agent-specific feedback, agents may receive feedback for contributions that are not theirs—making this feedback noisy and evolving using it, inefficient.



Deriving agent-specific feedback from a collective evaluation of the whole system poses the well-known *multiagent credit assignment problem*, which calls for accurately attributing performance outcomes to individual agents. Difference Evaluation (D) [1] is a prominent method that addresses this by estimating an agent's contribution by removing it from the system and re-evaluating the system's performance.

While the fine-grained and isolated feedback computed with D is beneficial in large systems performing complex tasks, D's reliance on re-evaluations of system performance may introduce too much of a computational overhead for small systems performing simple tasks. If the multiagent coordination task is simple and involves few agents, the general system feedback is often informative enough. It is when we consider systems and tasks of an in-between nature of size and complexity, that it becomes unclear whether fully isolating feedback with D is better, or simply learning from the general system feedback. We thus investigate the following question:

"How should a single agent's feedback be isolated from the system feedback to best suit the complexity of a problem?"

To this end, we generalise D so that an agent is also credited for the contributions of select other agents. Varying the number of agents whose contributions are included in an agent's feedback alters feedback specificity[1]. If a subset of the agents in the system all receive credit for each other's contributions, then this feedback need only be computed once for the subset—saving compute steps. We call this flexible extension D^{flex}, and it lets practitioners trade between feedback specificity and computational cost.

Our contribution in this paper is a preliminary presentation of the D^{flex} operator, and an implementation in a coevolutionary framework that learns a policy for each agent in parallel. We use the credit computed using D^{flex} as a *local fitness* for each agent to use in evolution. We also compare D^{flex} with D and learning from just the system feedback. Initial experiments in the multiagent rover exploration problem show that D^{flex} provides useful control over the feedback specificity vs. computational expense trade-off. Through D^{flex}, our goal is to articulate this trade-off clearly, and encourage the research community to consider it at time of developing multiagent learning algorithms.

2 Background

2.1 Multiagent Systems and the Credit Assignment Problem

The multiagent credit assignment problem is central in multiagent systems. It involves quantifying each agent's contribution towards system performance, and providing this measure as feedback for

[1] Including more agents reduces specificity, whereas including fewer increases it.

the agent to learn from. Such agent-specific feedback promotes learning efficiency, as it filters out the effects of other agents. In reinforcement learning, the credit for each agent's contribution to the system performance is provided as a reward [9, 11]. In evolutionary algorithms, credit is provided as a fitness value that evaluates an agent's policy [4, 5].

2.2 Related Work

Difference Evaluation. The Difference Evaluation operator (D) is a prominent approach for tackling the multiagent credit assignment problem [4, 8]. For an agent i, D_i is given by:

$$D_i = G(\mathbf{z}) \; - \; G\big(\mathbf{z_{-i}} \cup \mathbf{c_i}\big) \; [12], \tag{1}$$

where $\mathbf{z}$ represents the system's joint action (combined action of all agents), $G(\mathbf{z})$ denotes the global performance measure, $\mathbf{z_{-i}}$ is the joint action excluding agent i's input, and $\mathbf{c_i}$ is a counterfactual (or default) action that substitutes agent i's original action. The term $G\big(\mathbf{z_{-i}} \cup \mathbf{c_i}\big)$ estimates how the system would perform without agent i's contribution. Thus, D_i isolates the contribution of agent i to the performance of the whole system.

Difference Evaluation has demonstrated effectiveness in methods that employ Cooperative Coevolutionary Algorithms (CCEAs) to evolve multiagent joint policies, where it delivers a local fitness signal that quantifies each agent's individual contribution to the overall system performance [2, 4, 6].

3 Method: $\mathbf{D^{flex}}$

We now describe how to compute credit using $\mathrm{D^{flex}}$.

Preliminaries

- Let the multiagent team comprise k distinct agents, and the team policy π consist of k single-agent policies $\pi_1, \pi_2, \ldots, \pi_k$.
- Let each agent $i \in \{1, 2 \ldots k\}$ have a corresponding *credit set* $s(i) \subseteq \{1, 2, \ldots k\}$ such that agent i receives credit for the contributions of each agent in s.

$\mathbf{D^{flex}}$ Computation

We refer the reader to the original D equation (Equation 1). Now, for an agent $i \in \{1, 2 \ldots k\}$, given a credit set $s(i) \subseteq \{1, 2, \ldots k\}$, we define $\mathrm{D^{flex}}$ as:

$$\mathrm{D}_i^{\mathrm{flex}} = G(\mathbf{z}) - G(\mathbf{z_{-s(i)}} \cup \mathbf{c_{s(i)}}) \tag{2}$$

Here, we effectively take out the contribution of $|s(i)|$ agents from the system, and assign the difference in team performance as agent i's credit. On one extreme, if $s(i) = \{i\}$, this equation reduces to the original D equation (Equation 1). On the other extreme, if $s(i) = \{1, 2, \ldots k\}$, the second term in Equation 2 essentially becomes zero, meaning that the agent simply receives the whole team's performance, $G(\mathbf{z})$, as credit.

If, for an agent i, $s(i) = s(j) \forall j \in s(i)$, then each agent in the credit set $s(i)$ shares the same feedback, which means that the $\mathrm{D^{flex}}$ value (Equation 2) need only be computed once for the entire set. This saves $|s(i) - 1|$ re-evaluations.

The Cost of Evaluating System Performance

While $\mathrm{D^{flex}}$ requires the multiagent system to be evaluated both with the set of original constituent policies and counterfactual replacements, each such evaluation does not necessarily require a full episode-wide interaction with the environment. In many domains, agents may have access to a function that directly evaluates trajectories (state-action pairs of each agent), or predicts their performance [13]. Thus, instead of re-simulating or rolling out the policy in the environment, we may simply be able to construct counterfactual trajectories directly, and evaluate them. Such evaluations, although not fully reflecting of the decision-making that a policy would demonstrate in an actual rollout, are significantly cheaper, and dominated by actual policy rollouts in computational cost.

CCEA Implementation

We now set up a CCEA to evolve a parametrised multiagent control policy. Each subpopulation in the CCEA is tasked with evolving a single agent's parametrised policy. In each generation, a policy for each agent is randomly sampled from the respective subpopulation and arranged to form a multiagent team policy. This team policy then interacts with the environment and collects rewards for one complete episode. The cumulative reward from an episode is assigned as the team's evaluation. From the team evaluation, we then compute the credit for each agent using $\mathrm{D^{flex}}$, as described in Equation 2, and assign it as its policy's local fitness. We return each policy to its respective subpopulation, and sample a new policy from each subpopulation. We repeat this until each policy from each subpopulation has been evaluated and received a local fitness value. Each subpopulation is then evolved for one generation using selection (via binary tournament), crossover (via simulated binary crossover [7]), and mutation (Gaussian perturbation). We then repeat this procedure for several generations until convergence.

4 Experiment

The problem at large is that of multiple agents coordinating to perform a complex task, when the only feedback available evaluates the entire team and not individual agents. Within this context, we seek to test two effects: that of an agent receiving credit for actions of agents in addition to itself ("specificity effect") and that of the evaluations that are saved when multiple agents share credit ("sharing effect"). To study the specificity effect, we must vary how many agents' contributions a single agent receives as credit. To study the sharing effect, multiple agents should have identical credit sets, so that credit for these agents is also identical and thus need only be computed once.

4.1 Testing Domain

We use an instance of the Multiagent Rover Exploration (Rover) Domain, a classic multiagent coordination domain that has been in several previous works [1, 3]. In this domain, multiple rovers must coordinate and explore a 2-dimensional surface to capture Points of Interest (POIs) spread across the environment. Rovers receive a reward for each successful capture of a POI, but must avoid congesting a single POI, as a POI may only be captured once. Thus rovers must not only learn the navigational skills to approach

POIs, but also coordinate with other rovers to ensure they capture distinct POIs.

Each rover agent's policy is a fully connected feed forward neural network. Each agent's state input comprises four sensors pointed in the cardinal directions to sense POIs, and four sensors to sense other agents. Each sensor input is a sum of e^{-d} values, where d is the distance to the feature being sensed. From these readings, the agent must output d_x, d_y navigational actions.

We now define our instance of the Rover domain. It is of dimensions 25×25 units with 80 POIs, and each POI may be captured from a maximum of 0.5 units away. We arrange the POIs in an evenly spaced grid, with the exact centre of the map left empty. We test 8 agents in this domain, with the agents starting from the centre of the map.

4.2 Algorithms

We use D^{flex} to compute credit with two different specificities, and two different levels of sharing. At the start of each run, each agent is assigned a credit set, and the agent receives credit for the actions of all the agents in the credit set. Each agent's credit set stays fixed for the entire training process.

To study the specificity effect, we propose 2D^{flex} and 4D^{flex}. In 2D^{flex}, each agent receives credit for its own contribution, and that of one other agent (more specific). In 4D^{flex}, each agent receives credit for its own contribution, and that of three other agents (less specific). The credit set is determined via random selection at the start of each trial, and remains fixed throughout training We ensure that the credit set is distinct for each agent.

To study the sharing effect, we propose $\text{2D}^{\text{flex}}_{share}$ and $\text{4D}^{\text{flex}}_{share}$. Instead of each agent having a distinct credit set, we partition the system of agents into credit sets of two for $\text{2D}^{\text{flex}}_{share}$, and of four for $\text{4D}^{\text{flex}}_{share}$, respectively. Each agent in a particular credit set receives credit for contributions of the whole set. Thus, within each credit set, each agent shares credit, and this credit value need only be computed once for the whole set—saving evaluation steps (Section 3).

Finally, for comparison, we also test D, and training with just the global team feedback with no credit assignment (G).

5 Results and Discussion

In each chart, we plot mean values with a solid line and marker, with the lightly shaded region representing the standard error of the mean across three statistical runs with seeds 2024, 2025, and 2026.

5.1 When Evaluations Are Cheap

When agents have access to a trajectory evaluation function, a team policy need not be re-simulated to be re-evaluated. In such a scenario, the evaluations themselves are not a significant computational bottleneck compared to the actual simulations of a team policy. Therefore, in **Figure 1** we compare the mean of the highest fitness attained in each *generation* by each algorithm.

Observation: As expected, D, with its highly specific credit value promotes efficient learning among the agents, and is significantly better than learning with just the team feedback, G. What is interesting, however, is how variants of D^{flex} provide a clean gradation of performance between that of D and G. With each step increase in specificity, we notice a small performance bump (2D^{flex} and $\text{2D}^{\text{flex}}_{share}$ are both better than 4D^{flex} and $\text{4D}^{\text{flex}}_{share}$).

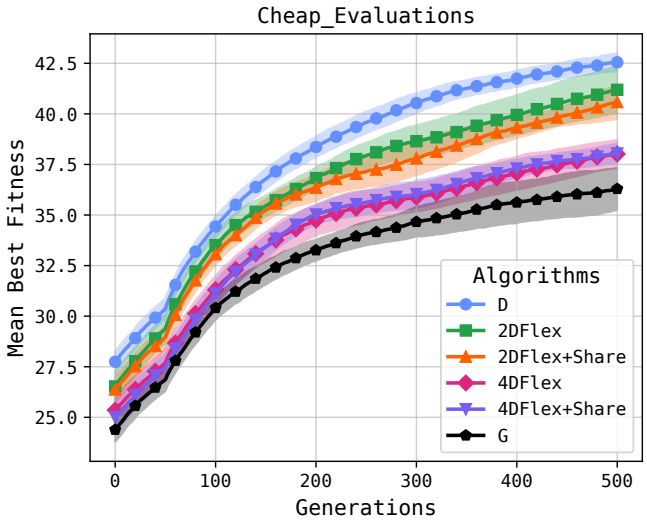


Figure 1: Mean best fitnesses at each generation by each algorithm. D^{flex} provides a gradation of performance according to credit specificity.

Explanation: Intuitively, this is the de-noising effect of increasing the specificity of feedback. As we assume evaluations are cheap, the credit sharing effect is absent, and the specificity effect is clearly highlighted.

Takeaway: When evaluations are cheap, it may be unnecessary to minimise them. Instead, leveraging these evaluations to the maximum possible extent—as done by D—is likely to yield the best performance. The specificity effect dominates the sharing effect, with more specific credit values being more desirable.

5.2 When Evaluations Are Expensive (i.e. Equivalent To Simulations)

In cases where no direct trajectory-evaluator function exists, each re-evaluation of a modified team policy requires a complete re-simulation. Studying the specificity vs. computational expense trade-off becomes crucial. In **Figure 2**, we compare the mean highest fitness attained in each *evaluation* by each algorithm.

Observation: G, requiring no extra re-simulations, significantly outperforms other methods. An interesting observation is that among methods that each compute a unique credit value for each agent (2D^{flex}, 4D^{flex}, and D), the more specific the credit, the better (D is the best among these, followed by 2D^{flex}, and then 4D^{flex}). Another observation is that for the same specificity, credit sharing offers significant performance boosts ($\text{2D}^{\text{flex}}_{share}$ is better than 2D^{flex}, and $\text{4D}^{\text{flex}}_{share}$ than 4D^{flex}).

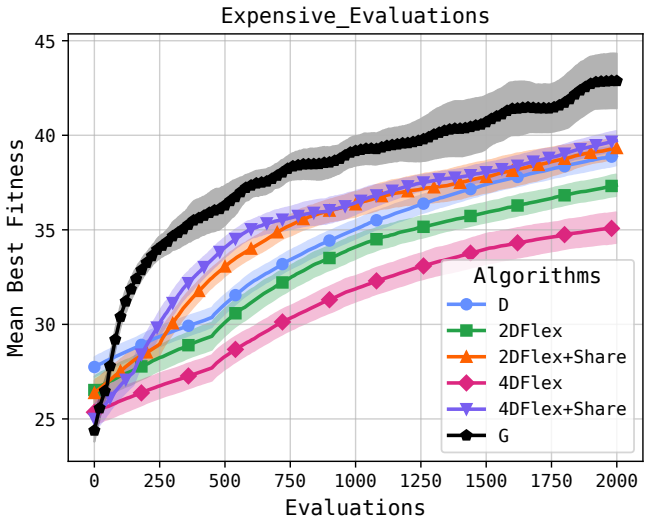


Figure 2: Mean best fitnesses at each evaluation by each algorithm. When evaluations are expensive, methods that rely less on them perform better. This is further highlighted by D^{flex} variants that use various number of evaluations for the same credit specificity.

Explanation: When re-evaluations are equally expensive as re-simulations, decisively acting on each evaluation becomes critical. Each algorithm performs selection, mutation, and crossover in each generation. However, for instance, G performs these after significantly fewer evaluations than, say, D. Therefore, for the same number of evaluations, more mutation steps, more selection steps, and more crossover steps help G evolve a better array of policy subpopulations than D. Similar logic applies when comparing variants of D^{flex}.

Takeaway: When evaluations are expensive, the credit sharing effect dominates the specificity effect. However, we are unable to confidently assert that the trade-off between specificity and computational expense is linear, given the close performance of $4D^{flex}_{share}$, $2D^{flex}_{share}$, and D. If the trade-off was linear, we would have clearly observed $4D^{flex}_{share}$ perform better than $2D^{flex}_{share}$, then followed by D.

6 Conclusion and Future Work

In this work, we presented an initial concept for D^{flex}, which exposes control over the specificity vs. computational expense trade-off in multiagent credit assignment. Our experiment in the multi-agent rover exploration domain further indicated the presence of this trade-off, and demonstrated D^{flex}'s ability to control it.

Much work must still be done to concretely describe the nature of this specificity vs. computational expense trade-off, however. For this, we will rigorously test variants of D^{flex} across team sizes, sizes of credit sets, degrees of reward sparsity, and under multi-objective settings. Additionally, we did not investigate how D^{flex} affects the evaluation of *indirect contributions* of agents, such as influencing other agents to perform rewarding actions, or being in their observations at crucial moments [10]. Therefore, would also like to take the first steps towards crediting such indirect contributions without explicit heuristics to measure them, by using D^{flex} instead.

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, Berlin, Heidelberg, 1–11.

[2] Adrian Agogino, Kagan Tumer, and Risto Miikkulainen. 2005. Efficient credit assignment through evaluation function decomposition. In *Proceedings of the 7th Annual Conference on Genetic and Evolutionary Computation* (Washington DC, USA) *(GECCO '05)*. Association for Computing Machinery, New York, NY, USA, 1309–1316. https://doi.org/10.1145/1068009.1068221

[3] Xinning Chen, Xuan Liu, Yanwen Ba, Shigeng Zhang, Bo Ding, and Kenli Li. 2023. Selective learning for sample-efficient training in multi-agent sparse reward tasks. In *ECAI 2023*. IOS Press, 413–420.

[4] Mitchell Colby and Kagan Tumer. 2012. Shaping fitness functions for coevolving cooperative multiagent systems, Vol. 1. 425–432.

[5] 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

[6] 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

[7] Kalyanmoy Deb, Ram Bhushan Agrawal, et al. 1995. Simulated binary crossover for continuous search space. *Complex systems* 9, 2 (1995), 115–148.

[8] Sam Devlin, Logan Yliniemi, Daniel Kudenko, and Kagan Tumer. 2014. Potential-based difference rewards for multiagent reinforcement learning. In *Proceedings of the 2014 International Conference on Autonomous Agents and Multi-Agent Systems* (Paris, France) *(AAMAS '14)*. International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 165–172.

[9] Jakob Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. 2018. Counterfactual multi-agent policy gradients. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 32.

[10] Everardo Gonzalez, Siddarth Viswanathan, and Kagan Tumer. 2024. Influence Based Fitness Shaping for Coevolutionary Agents. In *Proceedings of the Genetic and Evolutionary Computation Conference* (Melbourne, VIC, Australia) *(GECCO '24)*. Association for Computing Machinery, New York, NY, USA, 322–330. https://doi.org/10.1145/3638529.3654175

[11] Duc Thien Nguyen, Akshat Kumar, and Hoong Chuin Lau. 2018. Credit Assignment For Collective Multiagent RL With Global Rewards. In *Advances in Neural Information Processing Systems*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (Eds.), Vol. 31. Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2018/file/94bb077f18daa6620efa5cf6e6f178d2-Paper.pdf

[12] 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

[13] Golden Rockefeller, Shauharda Khadka, and Kagan Tumer. 2020. Multi-level Fitness Critics for Cooperative Coevolution. In *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems* (Auckland, New Zealand) *(AAMAS '20)*. International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 1143–1151.