Multi-Objective Credit Assignment for Multiagent Systems

Raghav Thakar[a,*], **Gaurav Dixit**[a] **and Kagan Tumer**[a]

[a]Collaborative Robotics and Intelligent Systems (CoRIS) Institute, Oregon State University
ORCID (Raghav Thakar): https://orcid.org/0009-0008-1156-9178, ORCID (Gaurav Dixit): https://orcid.org/0000-0003-4553-8405, ORCID (Kagan Tumer): https://orcid.org/0009-0007-3809-7257

Abstract. Multiagent systems are increasingly used in complex coordination tasks such as environmental monitoring, underwater exploration, and air-traffic management. These tasks often involve multiple, possibly conflicting objectives. Developing coordinated joint policies in such settings is challenging due to the difficulty in assigning credit to individual policies within a joint policy, especially when only the overall system performance is observed. This paper introduces the Multi-Objective Difference Evaluations (D_{MO}) operator, which enables credit assignment in multi-objective joint-policy evolution without scalarising the multi-objective reward. The D_{MO} operator measures the impact of an individual policy on the system's performance by comparing the hypervolumes of nondominated sets with and without the given policy. By preserving and promoting key policies, the D_{MO} operator enhances the evolution process, leading to more efficient joint-policy evolution. We conduct experiments in a continuous multi-rover exploration problem, highlighting the benefits of D_{MO} in achieving better coordination and performance in multi-objective multiagent systems.

1 Introduction

Complex tasks like environmental monitoring, extraterrestrial exploration, and air-traffic management are well-known applications of multiagent systems. Many such tasks involve not just one, but several, possibly conflicting objectives. For example, for an environmental monitoring task, such conflicting objectives may comprise 1) performing an evenly spread-out area coverage over the environment, 2) performing focused surveillance over particular points of interest, while 3) minimising the energy expense of each drone. To succeed in such tasks, multiagent systems must learn coordinated joint-policies that balance multiple objectives at once.

Learning good team-oriented joint-policies is challenging, particularly when the reward generated by the environment encapsulates the entire team's performance. This is the *credit assignment* problem, and is generally addressed by providing agent-specific feedback that aims to quantify a particular agent's performance. However, such approaches are not well suited for multi-objective settings, where an agent's impact needs to be assessed across multiple objectives.

Multi-Objective Optimisation approaches, especially population-based Multi-Objective Evolutionary Algorithms (MOEAs) provide a way to optimise for different trade-offs across the objectives by optimising for the Pareto front [35, 13, 19, 17]. On the other hand, credit assignment methods have been applied successfully in single-objective settings, ranging from Multiagent Reinforcement Learning (MARL) to Cooperative Coevolution [23, 14, 6, 3]. However, in multi-objective problems, like multi-objective MARL, credit assignment is typically addressed by scalarisation of the multi-objective reward using agent-specific utility functions [32, 20, 5, 21]. Prior work has explicitly tackled the multi-objective multiagent credit assignment problem, but also using a priori agent-specific reward scalarisation [33].

A drawback to using utility functions is the risk of enforcing sub-optimal preferences on agents and the need for substantial domain expertise to design them. Additionally, in many settings, a clear preference over the objectives may simply not exist beforehand. Population-based MOEAs avoid this problem of reward scalarisation, but do not fit with existing credit assignment techniques.

In this paper, we introduce the Multi-Objective Difference Evaluations (D_{MO}) operator to perform credit assignment in multi-objective joint-policy evolution without a priori scalarisation of the multi-objective reward. We use the difference in the hypervolume of a nondominated set of joint-policies with and without a joint-policy's constituent policy to measure that policy's impact on the system performance. D_{MO} is a multi-objective adaptation of Difference Evaluations (D), a state-of-the-art credit assignment operator [2, 3].

D_{MO} estimates agent-level credit without a priori scalarisation of the reward function, therefore capturing the true impact of individual agents in a multiagent system. This agent-level credit can be used to guide offspring-creation in the evolution process to produce potentially high-performing joint-policies. Population-based MOEAs can leverage D_{MO} to efficiently arrive at a true estimate of the Pareto front, where each Pareto-optimal joint-policy expresses a different trade-off among the objectives.

This work introduces and defines the D_{MO} operator. We then present a minimally-modified version of an existing MOEA, the NSGA-II algorithm, that leverages the D_{MO} operator and shows performance boost-ups in convergence-time to, and quality of, the Pareto front estimate. Lastly, we compare the performance of modified NSGA-II with NSGA-II and also perform an ablation study by replacing the D_{MO}-values with the global fitness[1] in the modified NSGA-II algorithm.

* Corresponding Author. Email: thakarr@oregonstate.edu.

[1] Please note that in this work, the terms 'reward' and 'fitness' are used interchangeably and refer to the numeric value generated by the environment for the multiagent team at the end of a simulation episode.

2 Background

2.1 Multi-Objective Optimisation

Many real-world problems are multi-objective, where improving performance in one area can negatively impact another (e.g., speed vs. safety vs. comfort in autonomous vehicles). Instead of seeking a single *best* solution, it is often preferred to develop a range of Pareto-optimal solutions, each offering different trade-offs across objectives. On the Pareto front (the set of Pareto-optimal solutions), no solution is *better* than another; all are considered equally optimal.

There has been substantial progress in recent years in multi-objective optimisation. A majority of the work generally falls into one of the two categories – methods that focus on 1) improving the Pareto front estimate [13, 19, 10, 17, 35], and 2) optimising a single *super* objective created by taking a weighted sum over all the objectives [22, 20, 5, 21]. In the context of multiagent systems, more agent-oriented versions of the latter, also referred to as utility-based approaches in the literature are generally favoured. However, these approaches can enforce sub-optimal preferences over objectives and can require substantial domain expertise to be developed. Enforcing preferences can also compromise the true multi-objective nature of the problem by optimising a predetermined choice instead of choosing from several optima.

2.1.1 Multi-Objective Evolution

If the end-goal is to achieve the best Pareto front estimate, population-based MOEAs provide a naturally-fitting paradigm for multi-objective optimisation. The inherent presence of multiple solutions in the population pool lends itself very well to the goal of arriving at multiple equally-optimal solutions [12].

The body of research on MOEAs is vast. Some representative work proposed algorithms such as PAES , PESA-II, NPGA, SPEA2 and NSGA-II [13, 19, 10, 17, 35]. A common feature among these algorithms are the *elites preservation* and *diversity preservation* mechanisms. By ensuring the preservation and proliferation of the best (elite) and most unique (diverse) solutions in each generation, these algorithms improve their speed of convergence with the Pareto front and the final spread of solutions [18]. Of these, the NSGA-II algorithm remains the most popular for problems with few objectives and serves as the most suitable for comparison. Some recent work also demonstrates neuroevolution of an agent in multi-objective settings, but instead of using diversity-preserving mechanisms, it adds diversity as an objective to also optimise for [29]. In this paper, the NSGA-II algorithm and its operators will be a common recurrence.

2.2 Multiagent Systems and the Credit Assignment Problem

One of the key problems in multiagent systems research is the credit assignment problem, where an agent must determine the effect of its actions on the system's performance. This quantified contribution must inform the agent's policy, promoting positive team-oriented actions. Credit can be provided in several ways, including as a reward in (deep) reinforcement learning, or as a learning signal to augment the selection probabilities of high-performing policies in evolutionary algorithms. Credit assignment becomes particularly important in environments with tightly-coupled tasks that require a simultaneous action by two or more agents to yield any reward. Without credit assignment, tasks requiring such explicit coordination may remain unsolved.

Existing credit assignment techniques like Difference Rewards [1], D++ [26], and even learning-focused methods like Value Decomposition Networks (VDN) [30], QMIX [27], and COMA [15] can be implemented with little to no modifications with agent-specific utility functions that scalarise multiple objectives into one. However, for getting a spread of Pareto-optimal solutions, this problem becomes considerably harder.

2.3 Difference Evaluations

The Difference Evaluation operator (D) can be used to estimate the contribution of a single agent to the entire multiagent system's performance [8, 3, 9, 2, 1, 14, 6]. For an *agent* i, the Difference Evaluation is defined as:

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

where $\mathbf{z}$ is the joint-action of the system, $G(\mathbf{z})$ is the global system performance, $\mathbf{z}_{-\mathbf{i}}$ is the joint-action of the system with the action of agent i removed, and $\mathbf{c_i}$ is a *counterfactual* action that agent i's action is replaced with. $G(\mathbf{z}_{-\mathbf{i}} \cup \mathbf{c_i})$ gives an estimate of the performance of a theoretical system without the contribution of agent i [26]. The *counterfactual* term $\mathbf{c_i}$ represents the notion of a *default* action with no contribution to the system performance. This default action comes intuitively in many problems. For example, the counterfactual term can represent a static agent that does not move from its initial starting position in a continuous environment-observation problem [24].

D has been successfully applied to tackle the credit assignment problem in approaches that evolve a multiagent system using Cooperative Coevolutionary Algorithms (CCEAs) by providing a local fitness evaluation to each agent [8, 3, 9]. In Multiagent Reinforcement Learning (MARL) approaches, D is used to provide granular feedback from the global team reward to condition the policy of each agent [14, 6]. Interestingly, D has also been applied in multi-objective MARL and multiagent NSGA-II [32, 33], but with the a priori scalarisation of the multi-objective reward.

3 Problem Domain

In this paper, we investigate the *continuous multi-rover exploration problem*. This domain serves as a proxy for several real-world multiagent problems, like environmental-monitoring, underwater-exploration, and distributed lunar sensing and measurement [11, 25, 31]. Several previous works have used a variant of this domain to test various multiagent coordination algorithms [26, 8, 3, 24]. In this paper, we describe a multi-objective version of the multi-rover exploration problem to test our method on.

3.1 Problem Description

The domain consists of a set of homogeneous rovers on a two-dimensional plane that must observe a set of stationary Points-of-Interest (POIs). POIs may have a *coupling* requirement, which would require multiple agents (rovers, in this problem) to *simultaneously observe* the POI for it to yield any reward. A POI is observed when $m \geq$ `POI.coupling` agents are simultaneously located within `POI.obs-radius` distance of the POI. The *observation window* of a POI is the window in an episode in which it can be observed. Even if the other requirements of the POI are met, the POI does not yield any reward unless the global timestep is within the observation window of the POI. The set of POIs can be divided into multiple categories, each of which, define a 'type' of POI. Each POI-type

can be made to possess different properties (such as position, reward value, coupling requirement, observation radius, observation window etc.), making the exploration and observation task more challenging. Sufficiently different properties among POI-types can necessitate the agents to demonstrate drastically different behaviours to be able to contribute to the global system reward. By tracking the rewards from each POI-type separately, we transform this problem into a multi-objective one, with each objective being to *maximise* the reward collected from each POI-type. The reward for observing a POI, its coupling requirements, its observation radius, its observation window, and its type are unknown to the agents, and they must coordinate to maximise the total system reward across each objective.

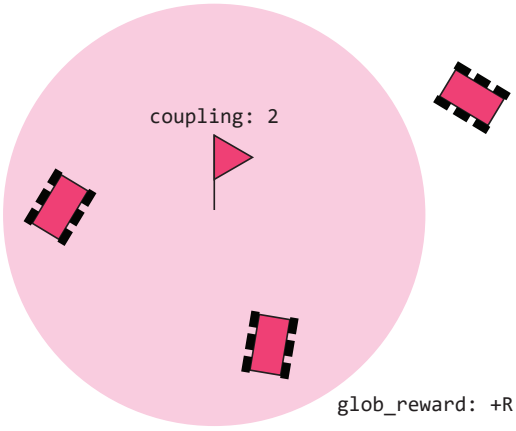


Figure 1: An example of a POI yielding a reward due to its coupling requirement being met through rovers observing it within its observation radius.

3.2 Agent Modelling

Each rover is equipped with three sets of sensors that provide information about its location, other rovers, and POIs respectively. The first set provides the global position of the rover in (x, y) form. The second set contains four sensors arranged planarly at 90° intervals around the rover to form four quadrants. Each sensor counts the number of other rovers in its respective quadrant. Lastly, the third set similarly contains four sensors that count the number of POIs in each quadrant. Each rover is constrained by the range within which its sensors can detect and count other rovers or POIs.

The state vector containing this sensor-information is the input to the agent's policy, represented as a fully connected feed-forward neural network. The policy network outputs the displacement in the x and y directions that the agent must take as an action. The displacement the agent can attain at each timestep is constrained to a reasonable limit respecting the environment dimensions

4 Method

In this section we provide a detailed explanation of the D_{MO} operator and its incorporation into a multi-objective evolutionary algorithm.

4.1 The D_{MO} Operator

EAs operate on a population of candidate solutions, wherein each *individual* in the population is a parameterised representation of the entity being optimised. When optimising, say, joint-policies for a multiagent system, each individual (or candidate solution) contains a vector of policies – each policy corresponding to an agent in the

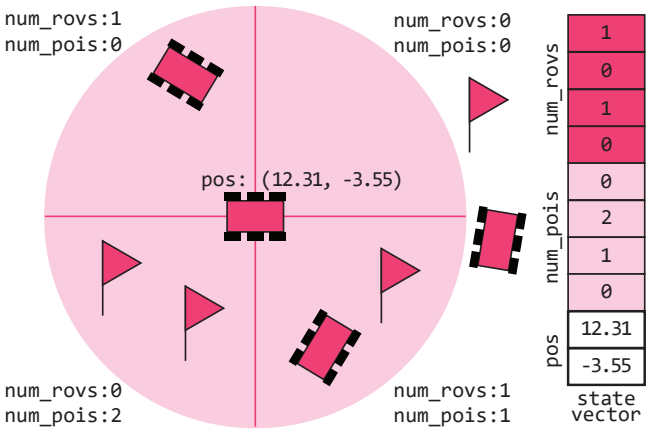


Figure 2: Sensor-layout on a rover. The rover counts the number of other rovers and POIs in each quadrant. Detection is restricted by the range of the sensors. Combined with the current position of the rover, this information makes up the input state vector for the rover's policy.

system. Existing MOEAs, like NSGA-II, operate on the *fitness vector*, containing a candidate solution's fitness on each objective. The application of NSGA-II in this context is sub-optimal, as it does not take into account the impact, or effect, of individual policies within a joint-policy (aka an individual) when performing the various evolutionary operators like *selection*, *crossover*, and *mutation* to create the *offspring* set. The lack of a policy-level (or agent-level) fitness means that highly impactful policies that may be a part of subpar joint-policies may be lost during evolution, while undesirable or low-impact policies that are a part of high-performing joint-policies may continue to proliferate. This potentially sub-optimal evolution can negatively impact the speed with which the evolution process converges, and the quality of the Pareto front estimate to which it converges.

The main challenge in assigning agent-level credit in multi-objective settings is finding a representative value that accurately captures the impact of a single policy on its joint-policy's performance. Before finding a representative value for a policy's impact, it is first important to make clear the features that comprise the 'performance' of a joint-policy:

- The magnitude of each fitness in the joint-policy's fitness vector
- To what degree the joint-policy increases the spread of its Pareto front
- How 'unique' the joint-policy is in the population of solutions

Our key insight here is that by measuring the contribution of a single policy to the *hypervolume* of the nondominated set its joint-policy lies on, we effectively measure the policy's contribution to each of the above three features, and thus to the true performance of its joint-policy. The hypervolume metric is accepted to capture all three of the above features by measuring proximity to the Pareto front, spread, and diversity [16, 34, 28].

To this end, we propose the Multi-Objective Difference Evaluation (D_{MO}) operator. In a population of joint-policies, D_{MO} computes the difference in the hypervolume of an individual's (aka joint-policy's) nondominated set, and the hypervolume of that nondominated set with one policy in the individual replaced with a default *counterfactual* policy. Intuitively, D_{MO} answers the question: *how does the hypervolume change, if a policy in an individual is replaced with a default?* The measure of this change is the policy's D_{MO}-value, or

impact. The D_{MO}-value is given by:

$$D_{\text{MO}}(j) = H(\mathcal{J}) - H(\mathcal{J}_{-J} \cup J^c) \tag{2}$$

where,

$$J^c = J_{-j} \cup j^c \tag{3}$$

and,

$$H(\mathcal{J}) = hypervolume(\{J.\texttt{fitness} \mid J \in \mathcal{J}\}) \tag{4}$$

Here, $j \in J$ is an individual policy in joint-policy (or individual) J. $\mathcal{J}$ is the set of joint-policies of the same nondomination-level as J in the population, i.e. it is the nondominated set that J is in. H computes the hypervolume of a nondominated set of joint-policies. $\mathcal{J}_{-J}$ is J's nondominated set with J removed, J^c is the *counterfactual* joint-policy that contains one *counterfactual* policy j^c inserted in place of a policy j, and J_{-j} is the joint-policy with j removed.

Assuming *maximisation* across all objectives, a positive D_{MO}-value means that the hypervolume reduced when that policy was replaced by a counterfactual one. The higher the D_{MO}-value for a policy, the more *impactful* it can be considered, and the more likely should be its preservation and proliferation in the evolution process.

The choice of the number of *parent* individuals used to create the *offspring* set is upto the System Designer. As long as each policy in the parent set of joint-policies has its D_{MO}-value computed and assigned, the offspring-creation step can leverage D_{MO}-values to create highly fit offsprings.

4.2 The Top-Level Evolutionary Algorithm

Having defined the D_{MO} operator, we now plug D_{MO} into an NSGA-II-like algorithm to showcase a practical implementation. Although the effects of D_{MO} are largely agnostic to the exact structure or type of multi-objective EA, in this paper, we elect to implement a minimally-modified version of the NSGA-II algorithm with the D_{MO} operator. We choose the NSGA-II algorithm for its immense popularity, and the high likelihood of familiarity with it among readers.

We utilise the `fast-nondominated-sort()` and `crowding-distance-assign()` functions as-is from the NSGA-II algorithm [13]. These two functions are responsible for the elite-preservation and diversity-preservation nature of NSGA-II respectively. The two main modifications we make to NSGA-II are as follows:

- D_{MO}`-assign()`: Compute and assign D_{MO}-values using Equation 2
- `make-`D_{MO}`-informed-offsprings()`: Create an offspring set on the basis of the D_{MO}-values

Algorithm 1 formally defines the top-level algorithm we use in this paper. Algorithm 2 is an algorithmic representation of the D_{MO}`-assign()` function that computes and assigns D_{MO}-values according to Equation 2. The `make-`D_{MO}`-informed-offsprings()` function creates an offspring set by utilising the policy-level D_{MO}-values. It performs selection of individual policies from the population based on their D_{MO}-values to assemble new offspring joint-policies. Algorithm 3 defines this process. The $select()$ function uses *softmax* to select a policy from a set of D_{MO}-values of policies. Figure 3 visually demonstrates this process, and also shows how the policy at a certain index in the offspring comes from selecting across the policy at the same index in each joint-policy in the population. This is unlike the offspring creation step in NSGA-II, which uses a binary-tournament selection using the nondomination-level and crowding distance, followed by a one-point crossover.

Algorithm 1 Top-level Evolutionary Algorithm

Initialise populations P_0, Q_0 each containing N joint-policies comprising K policies each
for $t \in [0, T]$ **do**
  $P_t = P_t \cup Q_t$
  foreach *joint-policy* $p \in P_t$ **do**
    `evaluate(`p`)`
  end for
  $F =$ `fast-nondominated-sort(`P_t`)`
  $i = 0$
  while $|P_{t+1}| + |F_i| \leq N$ **do**
    D_{MO}`-assign(`F_i`)`
    $P_{t+1} = P_{t+1} \cup F_i$
    $i = i + 1$
  end while
  `crowding-distance-assign(`F_i`)`
  `crowding-distance-sort(`F_i`)`
  D_{MO}`-assign(`F_i`)`
  $P_{t+1} = P_{t+1} \cup F_i[1 : (N - |P_{t+1}|)]$ // Discard excess
  $Q_{t+1} =$ `make-`D_{MO}`-informed-offsprings(`P_{t+1}`)`
end for

Algorithm 2 D_{MO}`-assign(`$\mathcal{J}$`)`

Input: A nondominated set $\mathcal{J} = \{J_0, J_1, ...J_{M-1}\}$ of joint-policies
$h_{\mathcal{J}} = H(\mathcal{J})$ // Hypervolume from Equation 4
foreach *joint-policy* $J \in \mathcal{J}$ **do**
  $\mathcal{J}_{-J} = \mathcal{J} \setminus J$
  foreach *policy* $j \in J$ **do**
    $J^c = (J \setminus j) \cup j^c$ // Swap j with a counterfactual policy
    $j.D_{\text{MO}}\texttt{-value} = h_{\mathcal{J}} - H(\mathcal{J}_{-J} \cup J^c)$
  end for
end for

Algorithm 3 `make-`D_{MO}`-informed-offsprings(`P`)`

Input: A population set P of N joint-policies, comprising K policies each
Output: An offspring set Q of N joint-policies
Initialise empty set Q
for $n \in N$ **do**
  Initialise blank joint-policy q
  for $k \in K$ **do**
    $q[k] = select(\{p.D_{\text{MO}}\texttt{-value} \mid p \in P\})$
  end for
  $Q = Q \cup \{q\}$
end for
return Q

5 Experimental Setup

5.1 Simulation Details

To show the effectiveness of using D_{MO}, we perform our experiments on instances of the multi-objective continuous rover exploration problem (rover domain hereon). We setup two environments

Property	E_{Easy}	E_{Hard}
Size	100×100 units	160×160 units
Episode Length	100 timesteps	240 timesteps
Number of Type A POIs	8	4
Number of Type B POIs	4	4
Type A Observation Radius	10 units	5 units
Type B Observation Radius	10 units	5 units
Type A Observation Window	100 timesteps	240 timesteps
Type B Observation Window	25 timesteps	40 timesteps
Type A Reward	+10, Repeats	+20, Does Not Repeat
Type B Reward	+30, Repeats	+30, Repeats

Table 1: Configuration of E_{Easy} and E_{Hard} environments. E_{Hard} is harder to collect rewards from compared to E_{Easy}. The greater environment size, smaller observation radii of POIs, and non-repeating nature of Type A rewards contribute to this increased difficulty. When a POI has a repeating reward, it incentivises agents to continuously observe it from the same position. However, when the reward does not repeat, it incentivises exploration of other POIs by the agents after being observed once, making reward-collection harder.

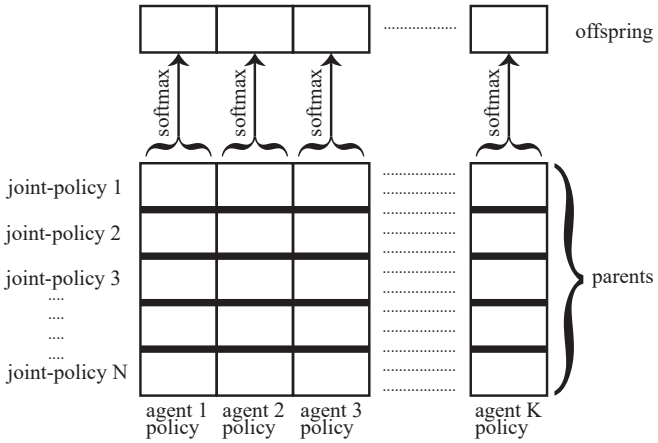
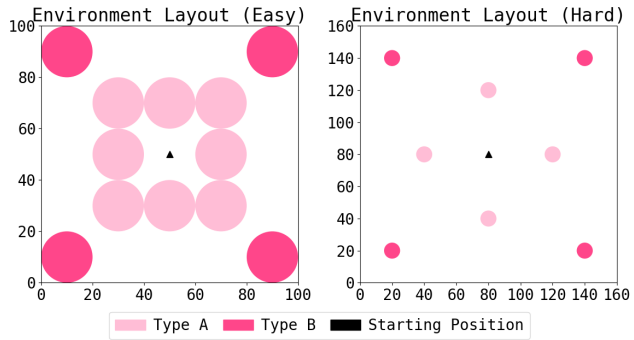


Figure 3: Offspring-creation by performing index-wise softmax over the D_{MO}-value of the policy at that index across all joint-policies in the parent set. Note how the offspring's policy at a certain index comes from the policy at the same index in the parent. Thus, an offspring joint-policy of size K may have up to K parents.



(a) E_{Easy} (b) E_{Hard}

Figure 4: Layout of POIs in E_{Easy} and E_{Hard} environments. The dense distribution of POIs in E_{Easy} allows teams to collect rewards even by demonstrating apparently *unintelligent* behaviours.

– E_{Easy} and E_{Hard} that differ in the spatial density of rewards that agents can collect. Each agent can displace by a maximum of 4 units at each timestep, and receives information from a 10 unit radius that makes up its state. In both environments, a team of 10 agents start from the centre of the environment and must appropriately *observe* POIs to yield a reward. The environments contain POIs of two types – *Type A* and *Type B* that represent two objectives. The team receives a cumulative reward for each objective at the end of the episode. Table 1 contains the exact configuration of both environments. Figure 4 shows the spatial layout of POIs in both environments.

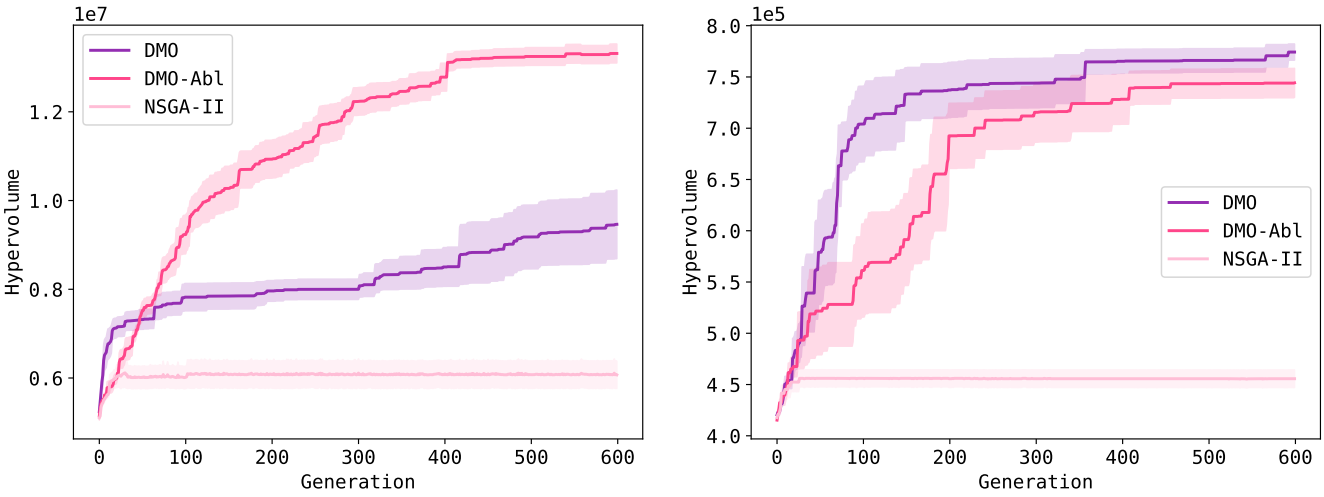
5.2 *Algorithmic Details*

We test three algorithms in the E_{Easy} and E_{Hard} environments – a modified version of NSGA-II that uses D_{MO}-values for credit assignment (D_{MO}), NSGA-II, and an ablated version of D_{MO} (D_{MO}-Abl) that simply uses the hypervolume (of the nondominated set a team belongs to) as agent-level credit instead of D_{MO}-values (which capture the difference in hypervolumes). Each algorithm is run for 600 generations with a population of 100 joint-policies containing 10 policies (one per agent) each. To begin with, each policy-network's weights are randomly initialised between $[-1, 1]$. In each generation, we create 100 offsprings. We mutate the offsprings by applying gaussian noise (Mean = 0, Standard Deviation = 0.05) to the weights of each policy-network of each offspring. For D_{MO}, the counterfactual policy fixes the corresponding agent to its starting position for the entire simulation episode. Each algorithm (D_{MO}, NSGA-II, D_{MO}-Abl) is run five times (each run of each algorithm with random initial populations and policy weights) to accurately analyse its performance.

6 Results

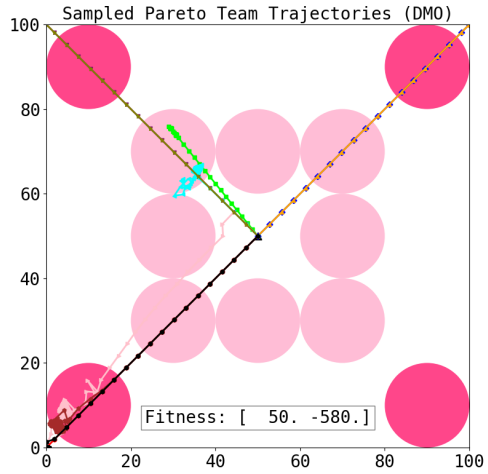
We study the quality of the Pareto front estimate evolved by each algorithm by comparing the hypervolumes at each generation in Figure 5. The solid lines represent the mean hypervolumes across multiple runs, while the shaded regions indicate the Standard Error of the Mean (SEM) around these means. Both D_{MO} and D_{MO}-Abl clearly outperform NSGA-II, which supports the intuition that it is highly inefficient to evolve good team-oriented policies without an estimate of the contribution of individual agents of the team. However, the performance comparison between D_{MO} and D_{MO}-Abl is more interesting.

In E_{Easy}, D_{MO}-Abl converges much faster to a Pareto front with a considerably larger hypervolume as compared to D_{MO}. In E_{Hard}, however, although small, there is a noticeable performance improvement when using D_{MO} as compared to D_{MO}-Abl. A plausible expla-

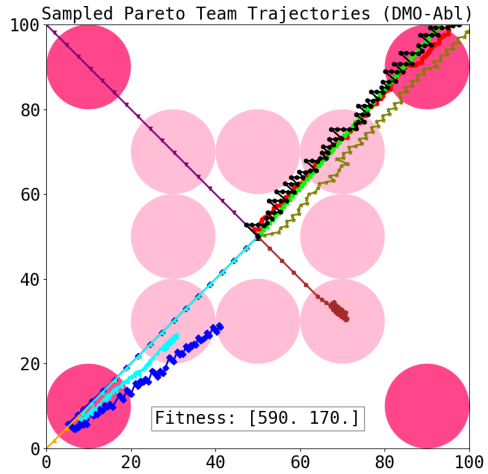


(a) E_{Easy} (b) E_{Hard}

Figure 5: Performance of D_{MO}, NSGA-II, and D_{MO}-Abl on E_{Easy} and E_{Hard} environments. As the environment becomes more challenging with sparser rewards, credit assignment becomes more useful in evolving good team behaviour.

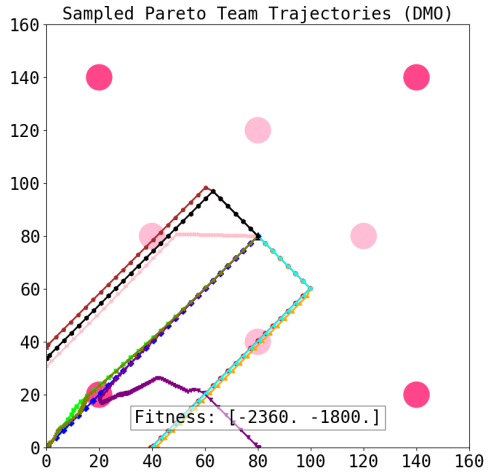


(a) D_{MO} in E_{Easy}

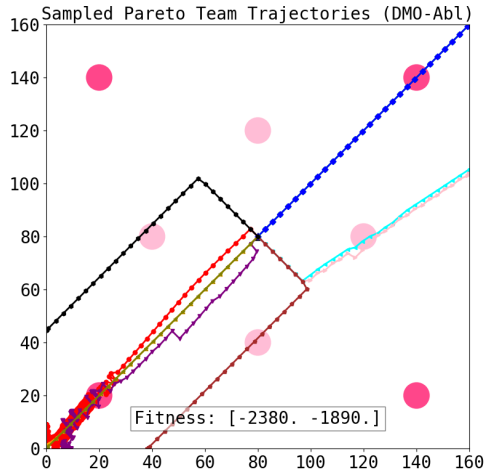


(b) D_{MO}-Abl in E_{Easy}

Figure 6: Trajectories from dominant joint-policies evolved using D_{MO} and D_{MO}-Abl in E_{Easy}.



(a) D_{MO} in E_{Hard}



(b) D_{MO}-Abl in E_{Hard}

Figure 7: Trajectories from dominant joint-policies evolved using D_{MO} and D_{MO}-Abl in E_{Hard}.

nation for why D_{MO}-Abl outperforms D_{MO} in E_{Easy} is the potential for being rewarded for several types of behaviours, as clear from Figure 4. The more *informed* credit estimated in D_{MO} is possibly too small a learning signal to quickly see performance improvements in the reward-rich E_{Easy} environment. Figures 6a and 6b, which show sample team trajectories achieved from D_{MO} and D_{MO}-Abl respectively, support this reasoning. D_{MO}-Abl is able to learn more diverse and coordinated behaviours that are potentially more impactful (spreading out to outer POIs, occupying inner POIs continuously, visiting several POIs for short durations, and having greater spatial coverage), while D_{MO} learns less diverse, and less 'developed' behaviours. The highly overlapping trajectories of multiple agents moving towards the extreme corners of the environment supports this. On the other hand, D_{MO}-values offer a visible benefit in the more challenging E_{Hard} environment (Figure 5b), likely due to requiring more intelligent behaviour to efficiently yield rewards.

Based on these results, an important general takeaway is that any form of agent-level credit, as long as it is roughly aligned with desired team behaviour, considerably aids the evolution process for multiagent teams and should feature centrally in algorithms for cooperative multiagent systems in multi-objective settings.

7 Discussion

In this paper, we presented the Multi-Objective Difference Evaluations (D_{MO}) operator to perform credit assignment in multi-objective evolution for multiagent systems. We applied D_{MO} to a minimally modified version of a popular MOEA, the NSGA-II algorithm, and showed significant performance boost-ups in performance on the rover domain. We then performed an ablation study to isolate the effects of the D_{MO} operator and demonstrate the importance of multi-objective credit assignment at large. Lastly, we qualitatively discuss the limitations of the D_{MO} operator and hypervolume indicator.

The proposed D_{MO} operator has been shown to perform well with the hypervolume metric. As a result, the drawbacks of using the hypervolume indicator transfer directly onto the D_{MO} operator. Its use in practical applications is highly sensitive to the choice of the reference point used to compute it. Specifically, the fitness of each individual in the population must dominate the reference point to be able to be included in any hypervolume calculation. This means that the 'worst' possible performance of an individual in the population must be known a priori, and must still be dominant over the reference point. Based on the problem, the lower bound for an individual's fitness may or may not be known beforehand, making the selection of the reference point challenging. The hypervolume metric is also expensive to compute, with the time complexity growing exponentially with the number of objectives [7]. There exist several other multi-objective performance indicators in the literature, like the Inverted Generational Distance (IGD) metric, the Spread metric, and the Epsilon metric [28, 4]. Based on the problem, one or more of these metrics may be more suitable to use with a D_{MO}-like operator.

To concretely assess the positive effects of the D_{MO} operator, we will conduct experiments with more agents and in more complex multi-objective domains as future work. Similarly, an ablation study at the joint-policy level (by measuring the impact of a whole joint-policy on the hypervolume, instead of a single policy contained in it) will be crucial to probe and study the impact of D_{MO} in multi-objective multiagent problems. Finally, to alleviate the shortcomings associated with the hypervolume indicator, we will consider additional performance indicators such as IGD and Spread [28, 4].

Acknowledgements

This work was partially supported by the National Science Foundation grant No. NSF IIS-2112633 and Air Force Office of Scientific Research grant No. FA9550-19-1-0195.

References

[1] A. Agogino and K. Tumer. Efficient evaluation functions for multi-rover systems. In K. Deb, editor, *Genetic and Evolutionary Computation – GECCO 2004*, pages 1–11, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg. ISBN 978-3-540-24854-5.

[2] A. Agogino and K. Tumer. Analyzing and visualizing multiagent rewards in dynamic and stochastic domains. *Autonomous Agents and Multi-Agent Systems*, 17:320–338, 10 2008. doi: 10.1007/s10458-008-9046-9.

[3] A. Agogino, K. Tumer, and R. Miikkulainen. Efficient credit assignment through evaluation function decomposition. In *Proceedings of the 7th Annual Conference on Genetic and Evolutionary Computation*, GECCO '05, page 1309–1316, New York, NY, USA, 2005. Association for Computing Machinery. ISBN 1595930108. doi: 10.1145/1068009.1068221. URL https://doi.org/10.1145/1068009.1068221.

[4] C. Audet, J. Bigeon, D. Cartier, S. Le Digabel, and L. Salomon. Performance indicators in multiobjective optimization. *European Journal of Operational Research*, 292(2):397–422, 2021. ISSN 0377-2217. doi: https://doi.org/10.1016/j.ejor.2020.11.016. URL https://www.sciencedirect.com/science/article/pii/S0377221720309620.

[5] T. Brys, A. Harutyunyan, P. Vrancx, M. E. Taylor, D. Kudenko, and A. Nowe. Multi-objectivization of reinforcement learning problems by reward shaping. In *2014 International Joint Conference on Neural Networks (IJCNN)*, pages 2315–2322, 2014. doi: 10.1109/IJCNN.2014.6889732.

[6] J. Castellini, S. Devlin, F. A. Oliehoek, and R. Savani. Difference rewards policy gradients. In *20th International Conference on Autonomous Agents and Multiagent Systems*, May 2021. URL https://www.microsoft.com/en-us/research/publication/dr-reinforce/.

[7] T. M. Chan. Klee's measure problem made easy. In *Proceedings of the 2013 IEEE 54th Annual Symposium on Foundations of Computer Science*, FOCS '13, page 410–419, USA, 2013. IEEE Computer Society. ISBN 9780769551357. doi: 10.1109/FOCS.2013.51. URL https://doi.org/10.1109/FOCS.2013.51.

[8] M. Colby and K. Tumer. Shaping fitness functions for coevolving cooperative multiagent systems. volume 1, pages 425–432, 06 2012.

[9] J. Cook, K. Tumer, and T. Scheiner. Leveraging fitness critics to learn robust teamwork. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '23, page 429–437, New York, NY, USA, 2023. Association for Computing Machinery. ISBN 9798400701191. doi: 10.1145/3583131.3590497. URL https://doi.org/10.1145/3583131.3590497.

[10] D. Corne, N. Jerram, J. Knowles, and M. Oates. Pesa-ii: Region-based selection in evolutionary multiobjective optimization. *Proc. 6th Int. Conf. Pparallel Prob. Solving from Nature PPSN-VI*, 01 2001.

[11] J.-P. de la Croix, F. Rossi, R. Brockers, D. Aguilar, K. Albee, E. Boroson, A. Cauligi, J. Delaune, R. Hewitt, D. Kogan, G. Lim, B. Morrell, Y. Nakka, V. Nguyen, P. Proença, G. Rabideau, J. Russino, M. S. da Silva, G. Zohar, and S. Comandur. Multi-agent autonomy for space exploration on the cadre lunar technology demonstration. In *2024 IEEE Aerospace Conference*, pages 1–14, 2024. doi: 10.1109/AERO58975.2024.10521425.

[12] K. Deb. *Multi-objective Optimisation Using Evolutionary Algorithms: An Introduction*, pages 3–34. Springer London, London, 2011. ISBN 978-0-85729-652-8. doi: 10.1007/978-0-85729-652-8_1. URL https://doi.org/10.1007/978-0-85729-652-8_1.

[13] K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE Trans. Evol. Comput.*, 6:182–197, 2002. URL https://api.semanticscholar.org/CorpusID:9914171.

[14] S. Devlin, L. Yliniemi, D. Kudenko, and K. Tumer. Potential-based difference rewards for multiagent reinforcement learning. In *Proceedings of the 2014 International Conference on Autonomous Agents and Multi-Agent Systems*, AAMAS '14, page 165–172, Richland, SC, 2014. International Foundation for Autonomous Agents and Multiagent Systems. ISBN 9781450327381.

[15] J. N. Foerster, G. Farquhar, T. Afouras, N. Nardelli, and S. Whiteson. Counterfactual multi-agent policy gradients. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence and Thirtieth Innovative Applications of Artificial Intelligence Conference and*

Eighth AAAI Symposium on Educational Advances in Artificial Intelligence, AAAI'18/IAAI'18/EAAI'18. AAAI Press, 2018. ISBN 978-1-57735-800-8.

[16] A. P. Guerreiro, C. M. Fonseca, and L. Paquete. The hypervolume indicator: Computational problems and algorithms. *ACM Comput. Surv.*, 54(6), jul 2021. ISSN 0360-0300. doi: 10.1145/3453474. URL https://doi.org/10.1145/3453474.

[17] J. Horn, N. Nafpliotis, and D. Goldberg. A niched pareto genetic algorithm for multi-objective optimization. volume 1, pages 82 – 87 vol.1, 07 1994. ISBN 0-7803-1899-4. doi: 10.1109/ICEC.1994.350037.

[18] W. Huang, Y. Zhang, and L. Li. Survey on multi-objective evolutionary algorithms. *Journal of Physics: Conference Series*, 1288(1):012057, aug 2019. doi: 10.1088/1742-6596/1288/1/012057. URL https://dx.doi.org/10.1088/1742-6596/1288/1/012057.

[19] J. Knowles and D. Corne. The pareto archived evolution strategy: a new baseline algorithm for pareto multiobjective optimisation. In *Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406)*, volume 1, pages 98–105 Vol. 1, 1999. doi: 10.1109/CEC.1999.781913.

[20] P. Mannion, K. Mason, S. Devlin, J. Duggan, and E. Howley. Multi-objective dynamic dispatch optimisation using multi-agent reinforcement learning: (extended abstract). In *Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems*, AAMAS '16, page 1345–1346, Richland, SC, 2016. International Foundation for Autonomous Agents and Multiagent Systems. ISBN 9781450342391.

[21] P. Mannion, S. Devlin, K. Mason, J. Duggan, and E. Howley. Policy invariance under reward transformations for multi-objective reinforcement learning. *Neurocomputing*, 263:60–73, 2017. ISSN 0925-2312. doi: https://doi.org/10.1016/j.neucom.2017.05.090. URL https://www.sciencedirect.com/science/article/pii/S0925231217311037. Multiobjective Reinforcement Learning: Theory and Applications.

[22] K. Miettinen and M. Mäkelä. On scalarizing functions in multiobjective optimization. *OR Spectrum*, 24:193–213, 01 2002. doi: 10.1007/s00291-001-0092-9.

[23] D. T. Nguyen, A. Kumar, and H. C. Lau. Credit assignment for collective multiagent rl with global rewards. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper_files/paper/2018/file/94bb077f18daa6620efa5cf6e6f178d2-Paper.pdf.

[24] A. Nickelson, N. Zerbel, G. Dixit, and K. Tumer. Shaping the behavior space with counterfactual agents in multi-objective map elites. In *Proceedings of the 15th International Joint Conference on Computational Intelligence - Volume 1: ECTA*, pages 41–52. INSTICC, SciTePress, 2023. ISBN 978-989-758-674-3. doi: 10.5220/0012164800003595.

[25] G. Notomista, C. Pacchierotti, and P. R. Giordano. Multi-robot persistent environmental monitoring based on constraint-driven execution of learned robot tasks. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 6853–6859, 2022. doi: 10.1109/ICRA46639.2022.9811673.

[26] A. Rahmattalabi, J. J. Chung, M. Colby, and K. Tumer. D++: Structural credit assignment in tightly coupled multiagent domains. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4424–4429, 2016. doi: 10.1109/IROS.2016.7759651.

[27] T. Rashid, M. Samvelyan, C. S. De Witt, G. Farquhar, J. Foerster, and S. Whiteson. Monotonic value function factorisation for deep multi-agent reinforcement learning. *J. Mach. Learn. Res.*, 21(1), jan 2020. ISSN 1532-4435.

[28] N. Riquelme, C. Von Lücken, and B. Baran. Performance metrics in multi-objective optimization. In *2015 Latin American Computing Conference (CLEI)*, pages 1–11, 2015. doi: 10.1109/CLEI.2015.7360024.

[29] J. Schrum and R. Miikkulainen. Evolving agent behavior in multiobjective domains using fitness-based shaping. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2010)*, pages 439–446, Portland, Oregon, July 2010. URL http://nn.cs.utexas.edu/?schrum:gecco10.

[30] P. Sunehag, G. Lever, A. Gruslys, W. M. Czarnecki, V. Zambaldi, M. Jaderberg, M. Lanctot, N. Sonnerat, J. Z. Leibo, K. Tuyls, and T. Graepel. Value-decomposition networks for cooperative multi-agent learning based on team reward. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, AAMAS '18, page 2085–2087, Richland, SC, 2018. International Foundation for Autonomous Agents and Multiagent Systems.

[31] M. Xanthidis, B. Joshi, J. M. O'Kane, and I. Rekleitis. Multi-robot exploration of underwater structures. *IFAC-PapersOnLine*, 55(31):395–400, 2022. ISSN 2405-8963. doi: https://doi.org/10.1016/j.ifacol.2022.10.460. URL https://www.sciencedirect.com/science/article/pii/S240589632202506X. 14th IFAC Conference on Control Applications in Marine Systems, Robotics, and Vehicles CAMS 2022.

[32] L. Yliniemi and K. Tumer. Multi-objective multiagent credit assignment through difference rewards in reinforcement learning. In G. Dick, W. N. Browne, P. Whigham, M. Zhang, L. T. Bui, H. Ishibuchi, Y. Jin, X. Li, Y. Shi, P. Singh, K. C. Tan, and K. Tang, editors, *Simulated Evolution and Learning*, pages 407–418, Cham, 2014. Springer International Publishing. ISBN 978-3-319-13563-2.

[33] L. Yliniemi and K. Tumer. Multi-objective multiagent credit assignment in reinforcement learning and nsga-ii. *Soft Computing*, 20(10):3869–3887, Oct 2016. ISSN 1433-7479. doi: 10.1007/s00500-016-2124-z. URL https://doi.org/10.1007/s00500-016-2124-z.

[34] E. Zitzler, K. Deb, and L. Thiele. Comparison of multiobjective evolutionary algorithms: Empirical results. *Evolutionary Computation*, 8(2):173–195, 2000. doi: 10.1162/106365600568202.

[35] E. Zitzler, M. Laumanns, and L. Thiele. Spea2: Improving the strength pareto evolutionary algorithm. 2001. URL https://api.semanticscholar.org/CorpusID:16584254.