Learning in Complex Domains: Leveraging Multiple Rewards through Alignment

Eric Klinkhammer
Oregon State University
klinkhae@oregonstate.edu

Connor Yates
Oregon State University
yatesco@oregonstate.edu

Yathartha Tuladhar
Oregon State University
tuladhay@oregonstate.edu

Kagan Tumer
Oregon State University
kagan.tumer@oregonstate.edu

ABSTRACT

In this paper, we introduce an alignment-based algorithm for improving the performance of reinforcement learners on problems where the reward signal cannot be collapsed into a single number. Many real world problems require an agent to balance performance, longevity, and safety, and do so across different timelines. The key to intelligent behavior in such complex domains is to enable the agents to determine "what matters when."

In this paper, we introduce the concept of local alignment that determines whether a sub-reward supports a global reward at a particular state and time. By selecting which policy to follow based on alignment, we show that agents using simple sub-rewards can combine their sub-reward-specific policies and learn a cohesive policy for complex coordination problems that agents trained on a single reward signal cannot learn.

KEYWORDS

Multiagent Coordination, Multi-Reward Learning

1 INTRODUCTION

Many real world problems can be formulated as a *state, action, reward* structure for reinforcement learning agents to solve. In order to do that, the original problem, with all its variabilities, intricate parts, and quirks is traditionally summarized into a single reward signal. For example, consider a team of soccer players. Each player must be able to pass and control the ball, shake off defenders, shoot the ball, etc. In such a scenario, each sub-task is critical to playing the game well. In the typical reinforcement learning structure, the performance of a learning agent is roughly summarized as a single reward signal, e.g. winning or losing the soccer match. Historically, learning algorithms applied to multiagent systems have used a single reward signal [1, 10, 15].

Problems arise when the task at hand has different conflicting or situationally-irrelevant sub-problems that may be relevant at a different state and time. In soccer, knowing how to score from a teammate's corner-kick is a very useful skill, but is not relevant for the majority of gameplay. An agent learning from a single reward signal is not likely to learn techniques like corner kicks by following the reward gradient; the win/loss reward is distributed over all their actions, and does not focus on learning a specific sub-task well. Such a reward signal can be very sparse and uninformative.

These issues with sparse reward signals becomes even more deceptive in multiagent problems, where an agent itself must be in the right state, trying to take the right action (e.g., get a header on their teammate's cross) while all their teammates are taking correct actions as well (kicking a cross properly, and clearing defenders). With a single sparse reward signal, the likelihood of the entire team exploring this particular state-action sequence in training is incredibly small.

Shaping the reward by dividing the goal task into intermediate objectives can help in learning by leveraging domain-specific information, and ameliorating the sparsity of rewards. This approach has been shown to improve the overall team performance in multiagent systems [9]. Transfer learning (TL) and multi-task learning (MTL) approaches have also been explored in multiagent domains to accelerate learning and generate better generalizable policies [2, 17, 25].

However, in all of these approaches, the performance of a learning agent is still summarized by a single reward signal, and learning in tightly coupled domains such as multiagent systems is difficult [21].

By splitting the reward signal into logical components, we can ascribe a separate reward for each component of the overall reward. This has a very human-like quality to it, where an agent would learn how to shake off defenders separately from practicing shootouts. This removes the noise in the reward signal caused by the interplay of all the sub-components, allowing an agent to focus on a single sub-reward. An agent learning to optimize only this sub-reward will learn an effective strategy for a subset of the original task. By learning policies for each sub-reward, an agent will be able to act optimally for any situation where the agent should optimize a particular sub-reward. This way, agents will have discrete sub-policies which are trained to solve a simpler problem than, for example, the entire game of soccer. These agents will have a collection of policies, each trained to optimize a single specific reward.

This formulation is similar to task decomposition [23], as we break apart the complex task into several sub-components. However, task decomposition does not consider time-varying tradeoffs between the sub-tasks. An agent equipped with discrete policies trained to optimize each sub-reward can only leverage them *if it knows which policy to use at a given state and time.* If the agent can identify when it finds itself in a corner kick situation vs a penalty shootout, it can use policies trained for these specific situations. A similar approach for switching between policies in multiagent adversarial scenario has shown to produce near optimal results [12].

This work was partially supported by NASA under grant No. 80NSSC18K0941

In general, we show that the most useful sub-reward *at a particular state and time* is the one most aligned with the original, coarse reward signal.

In order to find the most effective sub-reward, we have to be able to quickly compare each sub-rewards's alignment with the current state. By calculating the alignment between each reward, we can decide when we should follow each sub-reward to best approximate the global reward.

The key contribution of this work is to quantify the *alignment* among multiple reward functions spanning task components needed at different times, and use that alignment to determine *what matters when* to select the agents' actions when faced with complex tasks. The paper follows with relevant background in Section 2, the description and definition of alignment in Section 3 and Section 4. Experiments and results are found in Section 5 and Section 6, followed by a discussion of the results in Section 7 and concluding remarks in Section 8.

2 BACKGROUND

In this section we will introduce some relevant existing work that aim to solve complex tasks which are composed of multiple sub-objectives.

2.1 Transfer Learning

Transfer learning is the set of techniques used to improve the quality of learning and efficiency of resulting behaviors, by transferring the knowledge gained from one problem instance or domain to another. The idea is to learn a difficult task by starting on a simple task and progressively increasing the task difficulty. For example, in a multiagent coordination task, an agent can first be trained to coordinate with another single agent. Once the agent has learned this task, the learned knowledge can be transfered to a more complex scenario where the agent has to learn to coordinate with multiple other agents. The knowledge can be a direct transfer of information, such as via the values in a TD-based value table [24], or the weights in a neural network [26], or it can be transformed through some task-aware mapping [25]. Transfer learning shown some success in multiagent domains [2, 25]. However, assumptions such as full observability of the world [2], and availability of inter-task mappings for knowledge transfer [25] pose a hindrance for real world applications. Alignment guided control is similar to transfer learning in that both make use of different rewards with the end goal of improving the learning process for a final objective. However, they differ in that alignment does not require any transfer of knowledge. And unlike transfer learning, alignment switches between sub-policies based on the current *state and time.*

2.2 Multi-Task Learning

Multi-task learning (MTL) is based on the idea that generalization and performance on original task can be improved by leveraging the domain-specific information contained in training signals of related tasks [3]. For example in soccer, instead of specifically learning a policy for shooting the ball towards goal, MTL aims to learn a policy that can generalize across auxiliary tasks such as passing, or crossing, that involve some form of kicking the ball. MTL has been successful in machine learning applications such as natural language processing, speech recognition, computer vision and drug discovery [5, 8, 11, 22]. Generally in MTL, more than one loss function is being optimized. Alignment theory is different than MTL because the sub-tasks do not need to be auxiliary tasks. Instead of jointly learning policies for related tasks which benefit from shared information, alignment switches between rewards by focusing on what information matters now.

2.3 Multi-Objective Optimization

Multi-objective optimizations aims to find optimal solutions for tasks that are inherently measured by multiple objectives, either because different entities have conflicting goals, or because one entity must balance multiple criteria. There are two broad approaches to handling such problems: (i) scalarization, where the multiple objectives are collapsed into a single one using the trade-offs known to the designer; or (ii) Pareto concepts, where one seeks the points at which no objective can be increased without decreasing another [7, 27, 28]. However, linearly combining all objectives "freezes" the trade-off [4, 6, 13, 18] (that is, the exchange rate stays the same regardless of the supply and demand). Also, Pareto-based solutions treat each objective as an independent "entity" and do not allow the agent to worsen one objective even if doing so would provide some larger benefit to the mission goal. Thus multi-objective optimization fails to consider the "situational" need of a particular time and place.

2.4 Reward Shaping

Reward shaping aims to guide the agent's exploration to improve time to convergence by providing domain specific knowledge to the agent through additional rewards for intermediate objectives. For example, in a multiagent coordination task where multiple agents have to jointly observe POIs, a simple shaping approach would be to provide intermediate rewards for forming agent teams. Potential-based reward shaping ensure that good policies for a modified reward function are also good for the original [16], and has been shown to work in multiagent domains as well. Reinforcement learning with reward shaping has shown success in hardware validations of multiagent systems [14]; however success in tightly coupled multiagent tasks have been limited.

Another variant of reward shaping used in multiagent domains is the difference reward (D). The difference reward (D) for an agent is the difference between the actual global reward (G) and what the global reward would have been if that agent was not a part of the team [20]. Alignment and reward shaping are related concepts because the shaped reward function can be designed to be aligned with the reward for the actual goal task. However, reward shaping combines the intermediate rewards into a final single reward signal.

3 REWARD ALIGNMENT

Alignment is related to the concept of *factoredness* as defined by Agogino and Tumer [1]. An agent's reward signal (as a result of its action) is considered factored with respect to a global reward if the reward signal changes in the same direction as the global reward function's value, while holding everything else in the world fixed.

Traditionally, two reward signals are defined as being *aligned* if the rewards trend up and down together for all states in the signal.

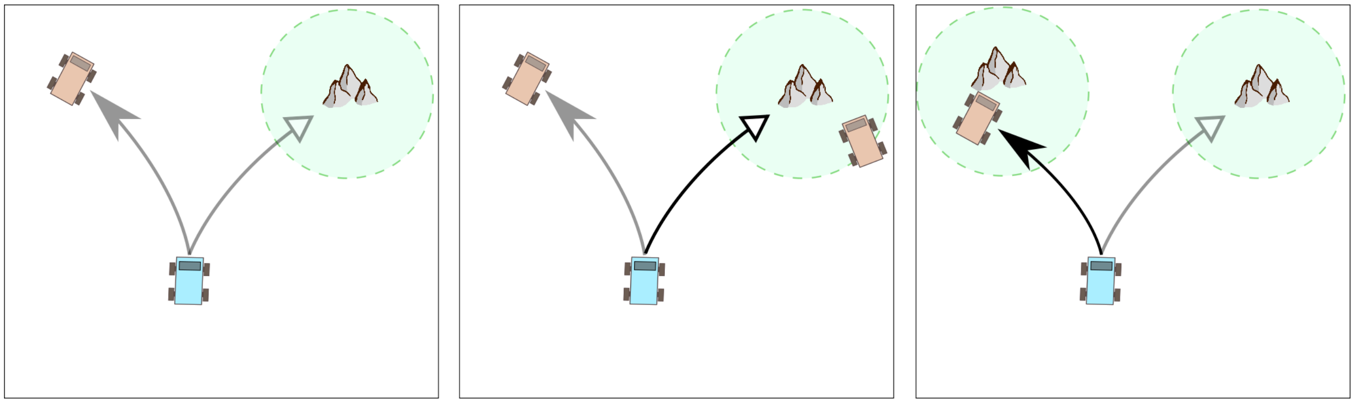

Figure 1: The exploring rover (blue, center) sees other rovers (red, left) and points of interest. The exploring rover has been trained to move toward other rovers and toward points of interest. However, the policies that move the rover toward a rover or a POI are exclusive: it will not try to navigate toward both at once. In this world, two rovers are need to be within the circle around the point of interest for the rovers to receive a reward. In the left scenario, there is no reward available to the exploring rover; there is no global reward for moving to the other rover, and no other rover is present at the point of interest. In the middle scenario, the agent will not receive a reward if it moves to meet the rover on the left. However, if it moves to go to the POI, it will enter the POI region with another rover and receive a score. Thus, the reward function to go toward a POI is aligned with the global reward. In the right scenario, the move-to-rover sub-reward is aligned with the global reward because it clearly moves the exploring rover into a POI region with another rover. The POI following reward is not aligned because it will direct the agent toward the POI on the right, where the agent will not receive any reward. In the middle and right scenarios, if our agent can identity the most aligned reward of the two it was trained with, it will be able to move and receive a reward.

That is, for a given agent, a local reward function R_{loc} is considered aligned with a global reward R_{glo} if:

$$\text{sign}\left(\frac{\partial R_{loc}(\mathbf{s},\mathbf{a})}{\partial \mathbf{a}}\right) = \text{sign}\left(\frac{\partial R_{glo}(\mathbf{s},\mathbf{a})}{\partial \mathbf{a}}\right) \tag{1}$$

where $\mathbf{a}$ is the action of the agent at state $\mathbf{s}$.

Given two reward signals which are not truly aligned, we can extend the definition to say they are *locally aligned* if they are aligned over the neighborhood of the state space. Figure 1 visualizes this concept of alignment.

To achieve a mathematical value for alignment, we compare the change in reward signal between two states when all actions are taken. The *alignment* between two rewards functions R_{loc} and R_{glo} on a subset of states ($\mathbf{S}_0$) will be given by Equation 2.

$$A_{\mathbf{S}_0}(R_{loc}; R_{glo}) = \frac{\sum_{\mathbf{s}\in\mathbf{S}_0}\sum_{\mathbf{a}}\sum_{\mathbf{a}'} u\left[(R_{loc}(\mathbf{s},\mathbf{a}) - R_{loc}(\mathbf{s},\mathbf{a}'))(R_{glo}(\mathbf{s},\mathbf{a}) - R_{glo}(\mathbf{s},\mathbf{a}'))\right]}{\sum_{\mathbf{s}\in\mathbf{S}_0}\sum_{\mathbf{a}}\sum_{\mathbf{a}'} 1} \tag{2}$$

In Equation 2, actions $\mathbf{a}$ and $\mathbf{a}'$ are valid actions from state $\mathbf{s}$, and $u[x]$ is the unit step function, equal to 1 if $x > 0$. This calculates a local alignment by measuring the change in reward by taking actions from the state $\mathbf{s}$. If an agent take various actions and the change in reward is the same, then it has found that the rewards are aligned for these actions. If at state $\mathbf{s}$, every action $\mathbf{a} \in A$ leads to an increase in the global reward and the local reward, the two rewards will be maximally aligned. Since the degree of alignment is based on the number of actions, two local rewards can be objectively compared by their alignment score if one reward function is aligned in more $(\mathbf{s},\mathbf{a})$ pairs.

By collecting several reward signals, an agent can find a good policy to solve the global reward signal through a composition of functions which are locally aligned in different subspaces in the broader state space. We can think of alignment as a linear combination of all sub-rewards into a single reward signal with the parameters of the linear combination continually adapting depending on the state of the world.

4 CALCULATING ALIGNMENT

In domains where the action set is well defined, finite, and reasonably sized, the alignment formulation in Equation 2 could be explicitly calculated at all states. However, in most complex real world problems this is not the case; it may not possible to explicitly evaluate which sub-reward is more aligned right now. Equation 2 can be approximated to get around this limitation. For example, in continuous action domains one can sample actions from the available action set; with a sufficient number of samples, the alignment scores of two reward functions will converge to their true value, and they can be compared appropriately.

4.1 Sampling to approximate alignment

To improve the tractability of calculating alignment for each agent, alignment values are stored as a mapping between agent state and alignment. Agents then compare their state during run time to the sampled alignment values and adopt the alignment of the most similar agent state. A k-dimensional (K-d) tree is used in these experiments because it supports nearest neighbor searches on average in $O(\log n)$ time. The process for generating the tree is described in algorithm 1. Each agent state is associated with only one alignment, with better (strictly positive changes in the target objective) overriding previous alignments. This one-to-one mapping within the data structure is why the algorithm uses an *upsert* operation instead of an *insert*.

Algorithm 1 Alignment K-d Tree Generation

```
1: function ADD_ALIGNMENT($tree, objA, objB$)
2:     $initW \leftarrow$ RAND_WORLD()
3:     $initA \leftarrow$ EVAL_OBJ($initW, objA$)
4:     $initB \leftarrow$ EVAL_OBJ($initW, objB$)
5:     $agentStates \leftarrow$ GET_STATES($initW$)
6:     for i ∈ Num Samples do
7:         $newW, dirs \leftarrow$ PERTURB_WORLD($initW$)
8:         $afterA \leftarrow$ EVAL_OBJ($newW, objA$)
9:         $afterB \leftarrow$ EVAL_OBJ($newW, objB$
10:        foreach $(state, dir) \in (agentStates, dirs)$ do
11:            if $afterA > afterB$ then
12:                UPSERT($tree, state, Policy_A$)
13:            else
14:                UPSERT($tree, state, Policy_B$)
15:            end if
16:        end for
17:    end for
18: end function
```

5 EXPERIMENTS

There are two major components to testing alignment. First, policies trained on a single sub-reward must be trained. Second, they must be tested as components of an agent selecting the most aligned sub-reward. These experiments take place in the simulated multi-rover exploration domain.

5.1 Rover Romain

The rover domain contains a set of rovers (agents) and POIs in a two-dimensional plane. Each POI has an observation radius. The agent's view is divided into four quadrants (north-east, north-west, south-east, and south-west), and the agent state is determined by 2 sensors (POI, Rover) in each quadrant which return the density of observable POIs and other rovers (respectively) in that quadrant. The density value is the sum of values of each of the POIs or rover divided by the euclidean distance from the sensor.

Thus at each timestep, the agent receives a feature vector of length 8 that summarizes the world from the agent's point of view. The agents select their actions during the training episodes using a feedforward neural network policy that takes the agent state as input and outputs a movement action (a (d_x, d_y) tuple). The reward for an episode is determined over the entire path of all of the agent's at the end of the episode and is then used as the fitness signal in the evolutionary algorithm training the agent policies.

Each POI must be observed simultaneously by three agents, otherwise no reward is received. This feature makes the task difficult

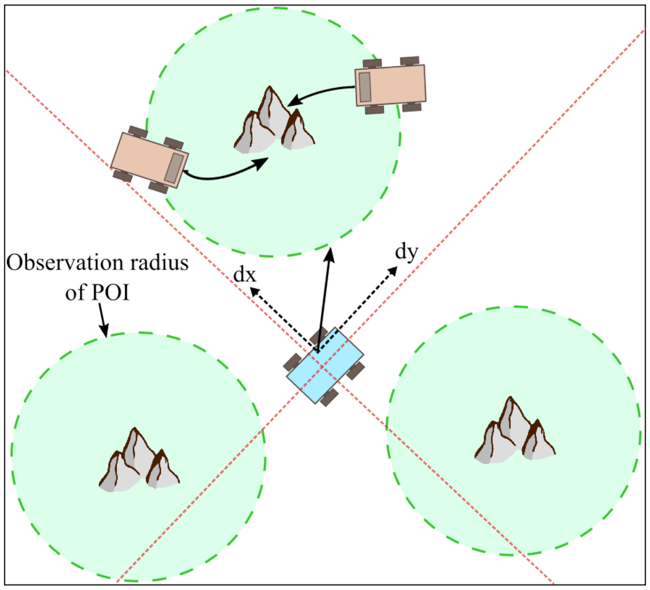

Figure 2: Diagram of the rover domain. The world is broken up into four quadrants relative to the rover position and orientation. POIs and fellow robots that are observed in each quadrant are summed resulting in 8 state variables. At each timestep, the robot's neural network controller yields two continuous outputs $[d_x, d_y]$, which determine the robot's motion in the next timestep. Each POI has an observation radius such that only robots within that radius are able to observe that POI and a coupling requirement such that no reward is received until that number of agents simultaneously observe the POI in a given timestep.

to learn because of how unlikely it is that multiple agents will stumble upon the proper configuration [21]. These features make the problem an ideal one to test alignment guided exploration because the additional objectives must add information for the method to work since there is no main objective gradient to rely on for much of the state space. An explanatory figure is provided in figure 2. The reward associated with each POI is its value, randomly determined at initialization to be between 4 and 10, divided by the average distance of the nearest 3 agents. The total global reward is:

$$P(\mathbf{z}) = \sum_p \sum_i \sum_j \sum_k \frac{V_p N^1_{p,i} N^2_{p,j} N^3_{p,k}}{\frac{1}{3}(\delta_{p,i} + \delta_{p,j} + \delta_{p,k})} \tag{3}$$

where $\mathbf{z}$ is the combined joint action of all agents, V_p is the value of observing the p-th POI, $\delta_{p,i}$ is the distance between the p-th POI and the i-th rover, and $N^a_{p,i}$ indicates whether POI i was the a-th closest rover within the maximum allowable observation distance δ_0.

$$N^1_{p,i} = 1 \text{ if } \delta_{p,i} < \delta_0 \text{ and } \delta_{p,i} < \delta_{p,l} \; \forall l \neq i \tag{4}$$

$$N^2_{p,j} = 1 \text{ if } \delta_{p,j} < \delta_0 \text{ and } \delta_{p,j} < \delta_{p,l} \; \forall l \neq i, j \tag{5}$$

$$N^3_{p,k} = 1 \text{ if } \delta_{p,k} < \delta_0 \text{ and } \delta_{p,k} < \delta_{p,l} \; \forall l \neq i, j, k \tag{6}$$

The rovers receive no information regarding the task prior to learning, and are trained with CCEA [19], a multiagent evolutionary algorithm described in Algorithm 2.

Algorithm 2 Cooperative Co-Evolutionary Algorithm (CCEA)

```
1:  foreach Generation do
2:      foreach Population do
3:          Generate $k$ successor solutions
4:      end for
5:      for $i = 1 \rightarrow 2k$ do
6:          Randomly select one agent from each population
7:          Add agents to team $T_i$
8:          Assign fitness to all agents based on simulation
9:      end for
10:     foreach Population do
11:         Select $k$ networks with rank probability
12:     end for
13: end for
```

5.2 Learning Sub-Rewards

As alignment is a property between objective functions, we compared a total of seven sub-rewards with the global reward function. These rewards fell into two categories: a reward to go toward POIs (GoToPOI) and variants of a pair of rewards to form teams (Shared Team and Exclusive Team). The rewards are detailed below. For each objective, we trained a policy on a simple world, with the experimental configurations detailed in Table 1. When using these trained policies with alignment or testing on a domain, the best policy from each team member's population was selected after a series of tests in random worlds. Random worlds are initialized by placing a cluster of agents in the center of the world, and randomly distributing the POI in around the agents so no agent is likely to be able to observe a POI without taking an action.

5.2.1 Single POI. A single agent placed randomly in a world with one POI. The agent is rewarded whenever it goes within the observation radius of the POI.

$$R_{single} = \sum_{a \in A} \sum_{p \in P} \frac{V_p}{\delta_{a,p}} \tag{7}$$

A is the set of all agents, and P is the set of all POIs within the observation radius of a. $\delta_{a,p}$ is the Euclidean distance between the positions of the rover and the POI.

5.2.2 Shared Teams. Populations of agents are placed in a world with POI who are rewarded for observing other agents. A shared team is defined as having another agent within a short radius of the other teammate, analogous to the observation radius for POI's in the rover domain. We use rewards for forming teams of 2, 3, and 4 agents. Below is the equation for a team size of 3.

$$R_{shared} = \sum_{a \in A} \sum_{i} \sum_{j} \frac{N^1_{a,i} N^2_{a,j}}{\frac{1}{2}(\delta_{a,i} + \delta_{a,j})} \tag{8}$$

A is the set of all agents. $N^i_{a,k}$ is an indicator variable for whether agent k is the i-th closest agent to agent a. $\delta_{a,k}$ is the distance between agents a and k. As before, the indicator variable is only non-zero when a, i, and j are distinct agents.

5.2.3 Exclusive Teams. Populations of agents are placed in a world with POI who are rewarded for forming explicit teams with others. A team is similar, but distinct from a shared team because an agent can only be in one team. A team is calculated by iteratively clustering the agents and seeing which tightly clustered agents are within an observation radius of each other. Agents in these teams are marked and removed from consideration, and the next best cluster of agents is determined. This continues until all agents uniquely satisfy the teaming requirements or are determined as not a part of a team.

Leaders are selected for teams based on descending leader score. Below is the equation for leader score (LS) of teams of size 3. The score is the individual component of R_{shared}.

$$\mathrm{LS}_a = \sum_{i} \sum_{j} \frac{N^1_{a,i} N^2_{a,j}}{\frac{1}{2}(\delta_{a,i} + \delta_{a,j})} \tag{9}$$

These leaders are, if not selected to be a member of another leader's team, the centroids of their respective teams. The reward is then calculated similarly to R_{shared}, except an agent cannot be a part of more than one team.

$$R_{exclusive} = \sum_{a \in A} \sum_{i} \sum_{j} \frac{M^1_{a,i} M^2_{a,j}}{\frac{1}{2}(\delta_{a,i} + \delta_{a,j})} \tag{10}$$

In this case, $M^i_{a,k}$ is only one if agent k is the i-th closest agent to agent a considering only agents not already selected to be in teams with higher scored leaders. A is iterated through in order of leader score.

$$M^i_{a,k} = 1 \text{ if } \forall a_i \in A \mid \mathrm{LS}_{a_i} < \mathrm{LS}_a, M^i_{a_i,k} = 0 \tag{11}$$

5.3 Alignment Demonstration

We first show that choosing a policy associated with the most aligned sub-reward produces logical decisions for an agent.

In the first experiment, an agent is placed inside a world which has two areas with other agents and POI, with a no man's land in the middle (similar to the choice faced in Figure 1). The agent is placed randomly around the middle, but with a bias to one side. On their left, a plethora of agents exist, but only one POI (which has two other agents next to it). On their right, a plethora of POI, but only one has agents next to it.

The agent has two sub-policies available to it: one trained to go towards POI, and the other trained to go towards other agents. The agent should be able to select the right policy among the assorted sub-rewards to score. This directly tests the "knowing what to chose when" aspect of the reward alignment selection, as agents should identify which reward function is more aligned and use the associated policy, as there is only one right decision in each case. (This is assuming only two rewards like GoTo POI and GoTo rover are provided.)

5.4 Rover Team Performance with Alignment

Next, we train teams of rovers in the general rover domain problem. We tested on several world sizes and configurations to show

Reward	Coupling	Observation Radius	World Size	Rovers	POIs	Timesteps
Single POI	1	4	30	1	3	25
Shared Team	1	4	40	2	0	20
Shared Team	2	4	40	3	0	30
Exclusive Team	2	2	40	6	0	20
Exclusive Team	3	3	40	9	0	25
Exclusive Team	4	4	40	12	0	30

Table 1: Training configuration for sub-reward policies. The coupling is the number of POIs (or agents) that a scoring agent must approach within radius units to receive a reward (scaled by distance). The world size is the length and width of a square world. All experiments were trained over 1000 generations with population of 40 neural networks per agent.

generalizable performance of alignment agents. Teams of 18 rovers are dropped into the middle of a 25 × 25 world. 6 points of interest are scattered around the team randomly, and each POI requires 3 rovers within the observation radius of 4 units to be marked off as observed. Agents have 30 timesteps to move around the world and attempt to observe the POI's.

For the teams using alignment, sub-policies were trained on random worlds with single sub-rewards described in Section 5.2.

We train additional teams of rovers in larger, more difficult domains. These tests put teams in a 50 × 50 world, with 45 timesteps to observe POI. One set looks at general performance in larger domains, and tests with 10 agents and 30 POI in the larger world. The next test highlights the performance-degrading effects of tightly coupled domains, putting 30 agents in the world with 10 POI. This highlights the effect of the tightly coupled reward because agents have a large space to jointly pick which POI to go toward. This creates much more room for error, as there are relatively fewer joint choices which give a positive reward. In both cases, the 3 agents are needed to observe a POI.

At each timestep, an agent using alignment will calculate the most aligned sub-reward and pick an action using a policy trained to operate on this sub-reward.

We compare alignment policy selection against a team of agents trained using evolutionary algorithm and transfer learning. The evolutionary algorithm trained agents using CCEA with the difference reward as the fitness function.

We also compared our results to a transfer learning method. Policies trained on simpler instances of the problem - a relaxed coupling requirement - were used to bootstrap training of the stricter coupling requirement instances. We used a three step process. First, policies were trained with CCEA with fitness determined by our main objective G, but with an observational coupling requirement of 1 (instead of 3). Then, those policies were used to seed the initial population in another round of CCEA trained on a fitness signal from the objective, but this time with an observational coupling requirement of 2. Finally, these policies seeded the training of the actual objective, observation with a coupling of 3.

As an additional baseline, we compare against random-action policies, which randomly select an action to take every timestep.

6 RESULTS

We show comparisons using the converged performance of agents using alignment policy selection and the other rewards. This is due

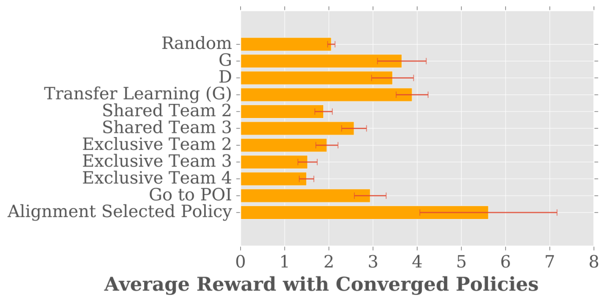

Figure 3: Converged performance for different learning policies in the tightly coupled rover domain. In this figure the best performance on the tightly coupled rover domain is achieved by teams using alignment policy selection to take actions. Due to the tight coupling constraints, teams trained difference rewards and sub-reward transfer learning do not find higher performance policies, and perform approximately as effectively as a policies trained on a single sub-reward.

to alignment policy selection being done in realtime; the method itself is not dependent on any learned criteria. Therefore, any changes in the performance of agents using alignment policy selection is due to the training of the underlying policies, not the alignment policy selection, and thus a learning curve between the two would not directly compare the performance of alignment policy selection with traditional methods. We test the general performance of agents aligning sub-rewards in a tightly coupled rover domain instance. The agents selecting policies based on alignment outperform agents trained on G, D, transfer learning with a curriculum of different couplings, and any of the individual rewards available to the alignment agents. This effect is seen in both the smaller, 18 agent, 25 × 25 world in Figure 3, and in the larger 30 agent, 50 × 50 world in Figure 4.

6.1 Alignment Selection Analysis

To support the claim that picking aligned rewards is what drives the improved performance in the tightly coupled domain, we examine the most aligned reward at every point in the world at a single time step. As the agent's position varies throughout the world, the most important component of their task changes. The most important

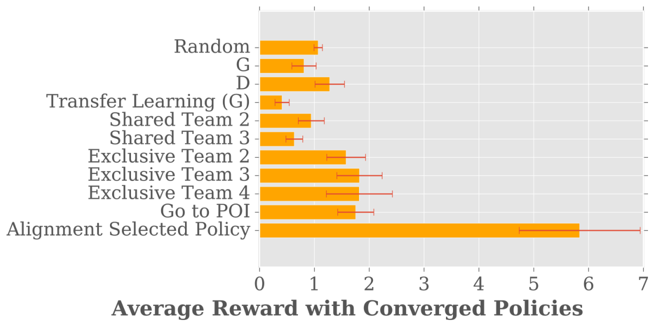

Figure 4: Converged performance for the tested methods in a 50x50 world with 10 POI, 30 agents, and 45 timesteps to observe. As the problem increases in size, the performance of alignment policy selection agents continues to hold strong against the baseline methods. Additionally, as the world size increases and the difficulty rises, the relative drop in performance between smaller, easier worlds and larger, difficult worlds is smaller for alignment agents. This drastically increases the relative performance of alignment agents versus the baseline, even though there is a drop in overall performance between the smaller worlds and this one.

task is captured by the most aligned reward signal, which we see in Figure 5.

In Figure 5, we show a map of the world with the type of the most aligned reward at each position overlaid. We see that alignment captures the most favorable reward signal to follow, as the GoToPOI reward brings the agent toward the scoring region in the lower left, and the Shared Team reward brings the agent to the scoring region in the upper right area. The Shared Team reward signal is ambiguous in the lower left due to the plethora of rovers in the area. The converse is true on the right side: the GoTo POI reward is ambiguous as it sees high rewards for moving toward any of the POI on the left.

We can draw intuition about this as due to the interactions between the two rewards on the periphery of the clusters. Note the cluster of aligned Team Forming (blue star) reward points in the lower-left corner, on the fringe of the aligned Single POI reward signal points (green dots). Here, following the Shared Team and GoTo POI rewards are roughly equal, as demonstrated by the few POI points intermixed in the outcropping. However, as the agent moves closer to the POI observation region it draws closer to the rovers. Critically, these rovers are not in the direction of the POI - the direction the agent should move toward in this situation. Instead, the agent moving toward the rovers will move along the border of the radius 4 circle that defines the POI observation region. If the other rover is just outside the POI region, an agent following the Shared Team reward will start moving *away* from the POI scoring region. Thus, it is more beneficial, and more strongly aligned, to follow the POI reward in this area.

7 DISCUSSION

Agents trained in simple cases, such as the single agent rover domain problem or the artificially constructed tightly coupled domain for a single agent show that combining policies trained on distinct sub-rewards is effective at creating a robust policy. Teams using difference rewards fail to learn an effective policy due to the tightly coupled nature of the problem. Agents trained with a transfer learning approach can build up knowledge about converging around POI, but only perform as well as some of the simple sub-reward policies applied directly on the rover domain.

Agents selecting sub-rewards based on their alignment with the global task objective show the highest performance. As seen in

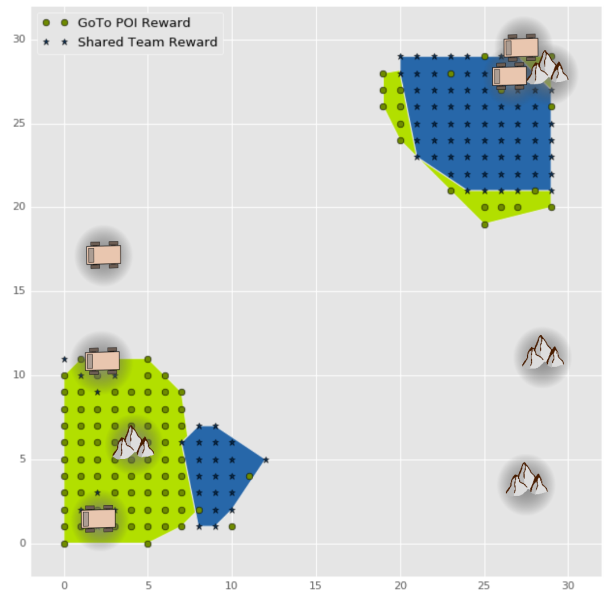

Figure 5: A rover domain problem with a coupling of 3 is presented to an agent. On the left, there are more rovers than POI, but these rovers extend beyond the observation radius of the lone POI. On the right, two rovers are near a POI, but multiple POI are found on the border. The agent has two rewards to choose from: a reward for going toward POI, and a reward for going toward other rovers. This situation is designed to test the agent in picking the obviously aligned reward wherever it may find itself in the world. Plotted are the local reward gradients for the POI reward (blue stars) and the rover reward (green dots). The areas shown are the most aligned reward at these points in the world, where the reward gradient and the gradient of G strongly match. The blank space is where G has no local gradient, so the each reward is ambiguously aligned and is not shown for visual clarity. As seen, the POI reward (blue stars) is correctly calculated as the most aligned reward in the situation at the bottom left of the world, as it homes the agent toward the region of high reward, whereas the rover reward does not guarantee moving the agent toward the POI observation area. The inverse of this situation is seen in the situation at top right of the world, with the Rover reward signal being calculated as most aligned in the presence of multiple POI.

Figures 3 and 4, this increase in performance is also not due to any specific sub-reward which efficiently solves the rover exploration task. Instead, a combination of these sub-rewards is required to achieve the increased performance.

Picking these sub-rewards by their alignment is effective because it allows the agent to answer the question *which reward matters when?* As seen in the analysis based on Figure 5, the agents calculating strongly aligned rewards are able to pick which sub-rewards solve the current state best. We define rewards with alignment values which are *positive* and *higher* for one reward than all others to be *strongly aligned.* This, in turn, provides them a structured manner to find the best action to take, which increases their overall performance.

By picking the most aligned policies and deducing an action from them, the agents are able to act as if they have a single policy which was trained to solve the original, complex problem.

One immediate disadvantage of this method is that the agents now need a policy for each sub-reward they will leverage to solve the task. As task complexity increases, the number of sub-rewards, and number of policies needed to train and remember, will increase. For robotics applications such as this, we do not foresee remembering policies becoming a direct inhibitor to solving the task due to the ever decreasing cost of computer storage.

The additional computation needed to train policies for each sub-reward may seem daunting, as now a variable number of neural networks are being trained to replace a single neural network. However, while using this alignment based policy selection does require training several sub-policies, the sub-rewards can be simple, and easier to learn than the general task as we have shown.

8 CONCLUSION AND FUTURE WORK

In this paper we introduce a method to tackle complex problems where traditional reinforcement learning summarizes a lot of problem information into a single reward signal. We first introduce a new take on reward alignment by looking at the *local alignment* of two reward signals at a given state. Then, by decomposing the original task into important sub-rewards and training policies on a single sub-reward, we create a set of policies which maximize performance on the sub-reward. Picking the most aligned sub-reward at the current state creates agents which focus on the salient features of a complex problem, so that they know *what* matters *when* as they move through the world.

This effect was demonstrated on the tightly coupled rover domain, as well as hand-constructed rover domain instances which tested the ability to correctly select aligned sub-rewards from the input state of the world.

Further work of this research will look at extending alignment learning to more domains to examine how to create sub-rewards for a single reward signal. Currently, sub-rewards are still hand-created and do not have a qualitative measure to differentiate a good sub-reward from a bad sub-reward. Furthermore, multiagent teams using alignment could generate human-level explanations of their actions by describing which aligned sub-reward is selected at a given time, and why it was the most aligned with G. Explanations like these could serve as a new research direction into assured autonomy and human robot interactions.

REFERENCES

[1] 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 (2008), 320–338. https://doi.org/10.1007/s10458-008-9046-9

[2] Olivier Buffet, Alain Dutech, and François Charpillet. 2007. Shaping multi-agent systems with gradient reinforcement learning. *Autonomous Agents and Multi-Agent Systems* 15, 2 (2007), 197–220.

[3] Rich Caruana. 1998. Multitask learning. In *Learning to learn*. Springer, 95–133.

[4] Carlos A Coello Coello. 1999. Comprehensive Survey of Evolutionary-Based Multiobjective Optimization Techniques. *Knowledge and Information Systems* 1 (1999), 269–308. https://doi.org/10.1007/BF03325101

[5] Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*. ACM, 160–167.

[6] Indraneel Das and John E Dennis. 1997. A closer look at drawbacks of minimizing weighted sums of objectives for Pareto set generation in multicriteria optimization problems. *Structural optimization* 14, 1 (1997), 63–69.

[7] Kalyanmoy Deb, Lothar Thiele, Marco Laumanns, and Eckart Zitzler. 2002. Scalable multi-objective optimization test problems. In *Evolutionary Computation, 2002. CEC'02. Proceedings of the 2002 Congress on*, Vol. 1. IEEE, 825–830.

[8] Li Deng, Geoffrey Hinton, and Brian Kingsbury. 2013. New types of deep neural network learning for speech recognition and related applications: An overview. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*. IEEE, 8599–8603.

[9] Sam Devlin and Daniel Kudenko. 2012. Dynamic potential-based reward shaping. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems-Volume 1*. International Foundation for Autonomous Agents and Multiagent Systems, 433–440.

[10] 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*. International Foundation for Autonomous Agents and Multiagent Systems, 165–172.

[11] Ross Girshick. 2015. Fast r-cnn. *arXiv preprint arXiv:1504.08083* (2015).

[12] Trong Nghia Hoang, Yuchen Xiao, Kavinayan Sivakumar, Christopher Amato, and Jonathan P. How. 2017. Near-Optimal Adversarial Policy Switching for Decentralized Asynchronous Multi-Agent Systems. *CoRR* abs/1710.06525 (2017). arXiv:1710.06525 http://arxiv.org/abs/1710.06525

[13] R. T. Marler and J. S. Arora. 2004. Survey of multi-objective optimization methods for engineering. *Structural and Multidisciplinary Optimization* 26, 6 (2004), 369–395. https://doi.org/10.1007/s00158-003-0368-6 arXiv:3-642-29068-X

[14] Maja J Matarić. 1997. Reinforcement learning in the multi-robot domain. In *Robot colonies*. Springer, 73–83.

[15] Risto Miikkulainen, Eliana Feasley, Leif Johnson, Igor Karpov, Padmini Rajagopalan, Aditya Rawal, and Wesley Tansey. 2012. Multiagent Learning through Neuroevolution. *IEEE WCCI* (2012), 24–46.

[16] Andrew Y Ng, Daishi Harada, and Stuart Russell. 1999. Policy invariance under reward transformations: Theory and application to reward shaping. In *ICML*, Vol. 99. 278–287.

[17] Shayegan Omidshafiei, Jason Pazis, Christopher Amato, Jonathan P How, and John Vian. 2017. Deep decentralized multi-task multi-agent rl under partial observability. arXiv preprint. *arXiv preprint arXiv:1703.06182* (2017).

[18] K. E. Parsopoulos and M. N. Vrahatis. 2002. Particle swarm optimization method in multiobjective problems. *SAC '02: Proceedings of the 2002 ACM symposium on Applied computing* 3 (2002), 603–607. https://doi.org/10.1145/508895.508907

[19] Mitchell A Potter and Kenneth A De Jong. 1994. A cooperative coevolutionary approach to function optimization. In *International Conference on Parallel Problem Solving from Nature*. Springer, 249–257.

[20] Scott Proper and Kagan Tumer. 2012. Modeling difference rewards for multiagent learning. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems-Volume 3*. International Foundation for Autonomous Agents and Multiagent Systems, 1397–1398.

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

[22] Bharath Ramsundar, Steven Kearnes, Patrick Riley, Dale Webster, David Konerding, and Vijay Pande. 2015. Massively multitask networks for drug discovery. *arXiv preprint arXiv:1502.02072* (2015).

[23] Peter Stone and Manuela Veloso. 1999. Task decomposition, dynamic role assignment, and low-bandwidth communication for real-time strategic teamwork. *Artificial Intelligence* 110, 2 (1999), 241–273.

[24] Matthew E Taylor, Peter Stone, and Yaxin Liu. 2005. Value functions for RL-based behavior transfer: A comparative study. In *Proceedings of the National Conference on Artificial Intelligence*, Vol. 20. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 880.

[25] Matthew E Taylor, Shimon Whiteson, and Peter Stone. 2007. Transfer via inter-task mappings in policy search reinforcement learning. In *Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems*. ACM, 37.

[26] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. 2014. How transferable are features in deep neural networks?. In *Advances in neural information processing systems*. 3320–3328.

[27] Eckart Zitzler, Marco Laumanns, and Lothar Thiele. 2001. SPEA2: Improving the strength Pareto evolutionary algorithm. *TIK-report* 103 (2001).

[28] Eckart Zitzler and Lothar Thiele. 1999. Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach. *IEEE transactions on Evolutionary Computation* 3, 4 (1999), 257–271.