AN ABSTRACT OF THE THESIS OF

Eric R. Klinkhammer for the degree of Master of Science in Robotics presented on June 7, 2018.

Title: Learning in Complex Domains: Leveraging Multiple Rewards through Alignment

Abstract approved: _____

Kagan Tumer

In this thesis, we introduce alignment-based algorithms for improving the performance of reinforcement learning solutions for 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."

We introduce the concept of local alignment as a metric that determines whether a sub-reward supports a global reward at a particular state and time. This work introduces three different approaches to using reward alignment for decision making, alignment policy selection, alignment action selection, and alignment based learning. By selecting which policy to follow based on this metric, 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.

In addition, we show that taking an action for which a reward is maximally aligned with a global reward outperforms learned policies and other methods. Instead of using sub-reward-specific policies or a global policy, agents select a direction in which a sub-reward most supports the global reward. We show that considering only aligned directions achieved rewards between 3 and 10 times better than reinforcement learning policies using the global reward.

Learning in Complex Domains: Leveraging Multiple Rewards through Alignment

by

Eric R. Klinkhammer

A THESIS

submitted to

Oregon State University

in partial fulfillment of
the requirements for the
degree of

Master of Science

Presented June 7, 2018
Commencement June 2018

Master of Science thesis of Eric R. Klinkhammer presented on June 7, 2018.

APPROVED:

Major Professor, representing Robotics

Head of the School of Mechanical, Industrial, and Manufacturing Engineering

---

Dean of the Graduate School

I understand that my thesis will become part of the permanent collection of Oregon State University libraries. My signature below authorizes release of my thesis to any reader upon request.

Eric R. Klinkhammer, Author

ACKNOWLEDGEMENTS

I would like to acknowledge my advisor, my lab, and my family.

TABLE OF CONTENTS

Page

1 Introduction 1
1.1 Learning for Long Term Autonomy 1
1.2 Reward Alignment Concept 4
1.3 Reward Alignment for learning in complex domains 6

2 Literature Review 10
2.1 Multi objective optimization 10
2.1.1 Pareto Optimization 11
2.1.2 Scalarization 12
2.2 Multi Reward Learning 14
2.3 Multi Task Learning 14
2.4 Transfer Learning 16
2.5 Reward Shaping 17

3 Concept 19
3.1 Reward Alignment 19
3.1.1 Calculating Alignment 21
3.1.2 Sampling to approximate alignment 22
3.2 Using Alignment for Decision Making 24
3.2.1 Following a policy of the most aligned reward 25
3.2.2 Selecting actions that are maximally aligned 25
3.2.3 Selecting aligned actions with learning 26

4 Rover Domain 28
4.1 Overview 28
4.2 Single Observation Required 30
4.3 Multiple Observations Required 31

5 Experiments 35
5.1 Learning Sub-Rewards 35
5.1.1 Single POI 37
5.1.2 Shared Teams 37

TABLE OF CONTENTS (Continued)

Page

5.1.3 Exclusive Teams 38
5.2 Alignment Demonstration 40
5.3 Multiagent Rover Performance with Alignment Decision Making . . . 40
5.4 Alignment Robustness 43
5.5 Learning from Alignment 43

6 Results 45
6.1 Reward Alignment Case Study Analysis 50
6.2 Reward Alignment Robustness to Reward Selection 53

7 Conclusion 58

Bibliography 61

LIST OF FIGURES

Figure Page

1.1 Different rewards are aligned in different regions of space and time. There are two rewards - one based on proximity to points of interest (mountains) and one based on proximity to other rovers. In this problem, two rovers 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. 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. Similarly, in the right scenario, the move-to-rover sub-reward is aligned with the global reward. 6

4.1 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. 29

4.2 The performance of rovers in the multiagent domain decreases as the number of agents required to make a successful observation increase. Reward shaping with difference rewards (D) trends in the same direction. 33

LIST OF FIGURES (Continued)

Figure | Page

6.1 Basic world - 25x25 with 18 agents and 6 POIs. Results show converged performance for different strategies in the rover domain with observations requiring three agents. In this figure the best performance is achieved by teams using alignment action selection with learning, followed by selection without learning, and finally alignment policy selection. Due to the high number of required observations, teams trained with 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. 46

6.2 More agent world - 50x50 with 30 agents and 10 POIs. Results show converged performance for different strategies in the tightly coupled rover domain. 47

6.3 More POI world - 50x50 with 10 agents and 30 POIs. Results show converged performance for different strategies in the tightly coupled rover domain. As the number of agents decreases, the importance of both forming teams and observing POIs successfully increases because it becomes much less probably that accidental teams are formed. For this domain setup, alignment based policy selection fails to outperform single objective teaming rewards. However, alignment based action selection still dominates the other methods. . . . 48

6.4 Large world - 50x50 with 30 agents and 30 POIs. Results show converged performance for different strategies in the tightly coupled rover domain. For this larger world, the global reward is able to perform better than many of the simple rewards, and within the margin of error of alignment based policy selection. However, alignment based action selection with and without learning still dominates the other methods. 49

6.5 The most aligned reward for an agent placed at various points in a world. Plotted are the local reward gradients for the POI reward (blue stars) and the rover reward (green dots). The vectors shown are the most aligned reward at these points in the world, where the reward gradient and the gradient of G strongly match. 51

LIST OF FIGURES (Continued)

Figure | Page

6.6 The performance of alignment based selection methods with and without the Exclusive Single POI objective (the most commonly selected objective). Alignment selection of direction achieved a substantially lower score, but alignment selection of policy did not have a statistically significant change. 54

6.7 The performance of alignment based selection methods without Exclusive Single POI, and without Exclusive Teams / Exclusive Team 4 (in addition to without Exclusive Single POI reward). 56

6.8 Alignment with learning. Alignment acts as training wheels for an agent training a neural network, with the neural network becoming progressively more responsible for action selection over the duration of training. Across all domains, alignment with learning is able to score equal to or greater than alignment for all domains but the 50x50 10 Agents 30 POIs. 57

LIST OF TABLES

Table Page

5.1 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. Single POI and Exclusive Single POI were trained identically. All experiments were trained over 1000 generations with population of 40 neural networks per agent. 39

6.1 Frequency an individual sub reward was selected as the most aligned. The majority of the time, the Exclusive POI reward is most aligned. These data were collected in a 25×25 world with 18 agents and 6 POIs over a period of 25 time steps. The frequencies are over 5400 actions. 53

6.2 Frequency an individual sub reward was selected as the most aligned with Exclusive Single POI removed. Without that reward, the frequencies are much more evenly distributed, with the exception of Single POI. These data were collected in a 25×25 world with 18 agents and 6 POIs over a period of 25 time steps. The frequencies are over 5400 actions. 55

6.3 Frequency an individual sub reward was selected as the most aligned with Exclusive Single POI and all Exclusive Team Forming rewards removed. These data were collected in a 25×25 world with 18 agents and 6 POIs over a period of 25 time steps. The frequencies are over 54000 actions. 55

6.4 Frequency an individual sub reward was selected as the most aligned with Exclusive Single POI and Exclusive Team Forming 4 rewards removed. These data were collected in a 25×25 world with 18 agents and 6 POIs over a period of 25 time steps. The frequencies are over 54000 actions. 56

LIST OF ALGORITHMS

Algorithm		Page
1	Alignment Calculation	23
2	Alignment K-d Tree Generation	24
3	Alignment-based policy selection	25
4	Alignment-based action selection	26
5	Cooperative Co-Evolutionary Algorithm (CCEA)	36

Chapter 1: Introduction

Robots are entering the world at an increasing rate, and as they do, the tasks they are solving become increasingly complex and multi-faceted. Traditional reinforcement learning methods use a single reward to capture, or at least attempt to capture, this complexity. Using multiple rewards to represent the different aspects of these problems introduces new problems - how should the different rewards be weighted, and how do robots deal with situations where what is important changes? We introduce alignment as a metric to identify which reward matters when. Alignment is a time-varying state-dependent weighting of rewards. Once the appropriate sub-reward is known, we introduce three decision making algorithms that make use of this information.

1.1 Learning for Long Term Autonomy

Robots have great potential to assist humans accomplish a wide variety of tasks in environments ranging from in the home to the ocean floor to the outer reaches of space. For robots to be useful outside of the controlled environments they are currently in, however, we must develop robots capable of acting autonomously for long periods of time. In these long missions outside of the research lab and far from the factory floor, robots must not only plan over exceptionally long time horizons,

but they must learn to accomplish new tasks and respond to unforeseen challenges, ideally by leveraging their existing knowledge of related tasks.

Despite the long time horizons and increasing problem difficulty, the problems associated with long term autonomy can still be formulated as traditional reinforcement learning problems with 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 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, 9, 18].

The use of a single reward signal, however, fails to fully capture all of the different components of the problem, e.g. the different skills a player must learn to win a match. Learning each task independently is problematic, however, because problems can arise when the task at hand has different conflicting or situationally-relevant sub-problems that vary in their importance across different states and time. To continue the soccer analogy, knowing how to score from a teammate's corner-kick is a very useful skill, but is not relevant for the majority of game play. An agent learning from a single reward signal is not likely to learn techniques such as corner kicks by following the reward gradient; the win/loss reward is distribute over all their actions, and does not focus on learning a specific sub-task well.

These rewards are not optimal for learning because they are often sparse, binary, and/or delayed. A sparse reward is frequently zero, making it difficult for the reinforcement learning to explore any actions that actually result in a reward. A binary reward gives no partial credit for partially correct actions, forcing the learning agent to be completely correct before a reward is given and removing a gradient of progressively more correct actions. A reward being delayed means that the reward for an action is not received immediately, but is given only after many time steps. Taken together, an agent not only must explore the correct action exactly in an action space of predominately useless actions, but also must properly attribute the reward to that particular correct action after taking many other (potentially zero reward) actions.

Beyond being difficult to use to learn these secondary, but still important, skills, such a sparse reward makes it difficult to know when to utilize niche skills. An agent might have learned to corner kick, but that information is not generally useful. Agents must be able to identify not only how to accomplish certain tasks, but when they are most appropriate in the greater context of winning the game.

The problems associated with using a sparse reward to learn - the difficulty in initially identifying correct behavior and the infrequent feedback - become even more challenging in multiagent problems, where the non-stationary aspect of the problem introduces added complexity. A non-stationary problem is time dependent and not Markovian (from the agent's perspective), with each subsequent state not being a strict function of the current state and the action taken. The introduction of other agents means that the reward an agent receives after taking the same

action in two identical states (assuming no noise) will not necessarily be the same - it depends on the actions of other agents. In order for an agent to learn, the 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 the correct actions as well (kicking a cross properly, and clearing defenders). With a single spare reward signal, the likelihood of the entire team exploring this particular state-action sequence in training is low.

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 [8]. 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, 36, 22].

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 [27].

1.2 Reward Alignment Concept

Reward Alignment is an effort to increase the signal from a sparse reward by intelligently examining and using sub rewards in the problem. By splitting the reward signal into logical components, we can ascribe a separate reward for each component of the overall reward. This problem solving technique has a very human-like

quality to it, where an agent would learn how to shake off defenders separately from practicing shootouts. This separation between discrete skills 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. Learning strategies for subsets of the original problem introduces a new problem: when a particular reward and corresponding strategy is relevant and should be followed. It is not enough to simply learn policies for each sub-reward, to act optimally an agent must also know which reward to follow when. Alignment provides a principled way to calculate which reward is relevant at a given time and place. This way, agents will have discrete sub-policies which are trained to solve a simpler problem than, for example, the entire game of soccer, but will be able to combine these sub-policies to solve the larger problem.

This formulation is similar to task decomposition [33], 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 scenarios has shown to produce near optimal results [12].

In order to find the most effective sub-reward, we have to be able to quickly

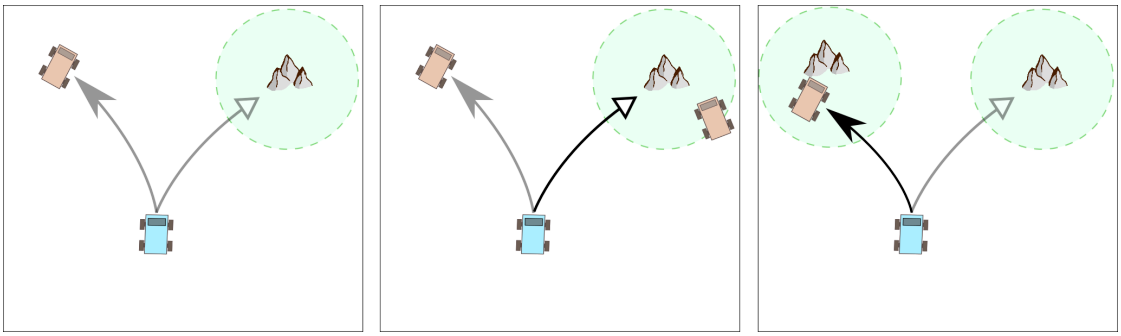

Figure 1.1: Different rewards are aligned in different regions of space and time. There are two rewards - one based on proximity to points of interest (mountains) and one based on proximity to other rovers. In this problem, two rovers need to be within the circle around the point of interest for the rovers to receive a reward. In the left scenario, there is no reward available to the exploring rover. 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. Similarly, in the right scenario, the move-to-rover sub-reward is aligned with the global reward.

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.

1.3 Reward Alignment for learning in complex domains

This work introduces three different approaches to using reward alignment for decision making: selecting the action which has the highest alignment without learning (alignment action selection), following the policy trained with the reward which has the highest alignment (alignment policy selection), and combining selecting the action which has the highest alignment with learning (alignment based learning).

Alignment policy selection uses the alignment of all subrewards in a particular state across a sample set of actions to determine the most aligned subreward, and then acts according to a policy designed to maximize that subreward. Figure 1.1 visualizes the alignment metric between two simple policies. The exploring rover has been trained on two rewards 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 problem, two rovers 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 take the correct action and receive a reward.

Alignment action selection considers the alignment of all subrewards for the same particular state and sample set of actions, but instead selects the action that resulted in maximum alignment for one of the subrewards. Alignment based

learning is a variant of transfer learning that generalizes from alignment action selection by training a policy in tandem with alignment action selection. Agents initially take only aligned actions, but gradually use their own policies to select actions instead.

Alignment is a useful metric because using if a reward is aligned with another in a direction, moving in that direction will optimize both in that direction. If it is known that two rewards are aligned but only how to optimize one of the rewards, then optimizing the reward that we do know how to optimize will, at the very least, not negatively impact the other reward. All strategies using alignment exploit this fact, and seek to optimize rewards with known optimization strategies in a manner that also improves another target reward. Alignment is a metric to determine if maximizing a reward can act as a surrogate for maximizing another. Alignment provides a way to leverage simple rewards and simple policies to solve complicated problems. This strategy can be used when the complicated problem cannot be solved with reinforcement learners and a single reward, or when there is insufficient data to learn a solution to that particular problem. Consider the use case of learning for long term autonomy; novel tasks must frequently be learned, often with very few samples. However, such a robot has already learned many other tasks previously, and would have access to both their associated reward functions and policies. Alignment provides a principled mechanism to combine simpler policies to solve more complex and novel tasks without needing to learn directly on the more complicated (and potentially unique) task.

By using alignment in a complicated multiagent domain, we show that these

methods of reward decomposition and using alignment decision making outperforms state of the art multiagent learning methods.

Chapter 2: Literature Review

We introduce the current literature in multi reward learning, including multi task learning, scalarization, and Pareto methods. Additionally, we introduce other methods of learning in complex domains, including transfer learning and reward shaping.

2.1 Multi objective optimization

Pareto methods and scalarization are two algorithmic approaches for optimizing multiple rewards. Pareto methods search directly in the multi dimensional reward space for a solution along the Parteo front, while scalarization maps multiple rewards to a single reward in order to subsequently use traditional optimization techniques. Alignment is not a Pareto-based method because the performance on subrewards is not important at all times - only the main reward is important. The final solution is not looking to maximize any reward but the global reward. The (time and state-dependent) willingness to sacrifice any particular reward without consideration for the trade-offs between any reward but the global separates alignment from these multiobjective optimization approaches. A solution alignment finds (the maximal global reward) is not even necessarily in the non-dominated Pareto set because the solution can be dominated by another solution with a

higher subreward value.

2.1.1 Pareto Optimization

When the learning goal is to optimize multiple objectives, a frequent difficulty is that the multiple objectives often conflict. For example, a surveying robot might have to balance the utility of the information gathered with the cost of the exploration [3]. A common strategy for a multi-objective problem is to find the Pareto Front [16][42][41], the set of combinations of objective values that are not strictly worse than any other.

The Pareto front is the set of solutions that are not strictly worse than any other solution. Informally, we might not be able to decide if its worth giving up wine for more cheese without knowing an individual's preferences, but if we can get more wine without giving up any cheese, that's a better choice. More formally, a solution point $\mathbf{x}$ is along the Pareto front if it is not dominated by any other point. A solution $\mathbf{x}^*$ dominates a solution $\mathbf{x}$ if:

$$\forall i \; \mathbf{x}_i^* \geq \mathbf{x}_i \wedge \exists i \; \mathbf{x}_i^* > \mathbf{x}_i \rightarrow \mathbf{x}^* \succ \mathbf{x} \tag{2.1}$$

Which one of these objective combinations is preferable will be determined by some weighting of the different objectives. When the combination function is additive or monotonically increasing, selecting between different combinations is a matter of preference along the Pareto Front. Under a nonlinear scalarization functions, the Pareto Front is more difficult to find. While the Pareto Front contains

the optimal solution, not only is it difficult to calculate, but there is no guarantee the solution is feasible for a generic nonconvex relationship between objectives.

Pareto methods look to find an optimal solution to a multi objective problem by finding a solution along the Pareto front. There are many different strategies for identifying solutions, and many approaches are evolutionary, including Multi-objective evolutionary algorithms based on decomposition [39], strength Pareto evolutionary algorithm [41], and non-dominated sorting genetic algorithm II [6]. It is not necessary to solve the full problem, and in fact Roijers shows that the convex coverage set is sufficient [29]. For non-convex problems, there are solutions, including Pareto Concavity Elimination Transformation (PaCcET) [37] - a transformation into a convex solution space, that can be used.

Alignment, however, is solving an easier problem. Instead of looking to optimize multiple objectives, it only optimizes a single one. The other objectives are only relevant along the path towards finding the solution.

2.1.2 Scalarization

The simplest way to solve a multi-reward problem is to convert the vector of rewards into a single reward, and then optimize that with all of the standard optimization tools. Scalarization is also the technique by which different points in the non-dominated set are compared with each other.

Scalarizing assumes the existence of a function $f : \mathbf{R}^n \times \mathbf{R}^n \to \mathbf{R}$ and weight vector w such that $f(x, w)$ allows two vectors x_i and x_j to be compared.

The most common scalarization function is a linear weighting of the objectives:

$$f(x, w) = \sum_{i=0}^{|w|} w_i * x_i \tag{2.2}$$

Less common is a geometric function, although Qiao et al. have shown that such a weighting is capable of solving many Pareto optimization problems [26].

$$f(x, w) = \prod_{i=0}^{|w|} w_i * x_i \tag{2.3}$$

Both linear and geometric approaches are convex functions, and the challenge is identifying the correct weight vector. Roijers proposes two motivating scenarios and corresponding solutions for which weight vector to use [28]. The first is when the weight vector will be known at run time (either calculated or determined from user preference). The approach here is to solve for solutions along the Pareto Front and select the correct one, and has been extended to at least fifty distinct objectives [13]. However, not all f are additive. In this case, even if the weight vector is known during planning or learning, selecting the proper action during execution is still a challenge.

Alignment can be viewed as a nonlinear scalarization function that is also a function of the state and time. In other words, the weighting of different objectives is dynamic during execution.

2.2 Multi Reward Learning

There are two main use cases for multiple rewards that are conceptually distinct. The first is that many problems cannot be expressed adequately with a single reward. The second is that multiple rewards can be used to learn multiple tasks simultaneously, both to improve the performance of a main task with auxiliary tasks and to transfer knowledge and learn faster across tasks. In practice, optimizing over multiple rewards can be the goal or the means to the goal. The use of multiple rewards is the focus of the this thesis, especially with regards to proposed research, but background is provided on the multi-objective literature.

Learning with multiple rewards can be a tool to improve the learning process. In Multi Task Learning (MTL), multiple tasks are trained simultaneously, sharing some aspect of their model (for example, two neural networks may share bottom layers). This technique can be used to train separate policies for separate tasks, but it can also be used to leverage an auxiliary task to augment the learning process for a main task. In this case of MTL, additional rewards - even if not related directly to the main task - improve learning.

2.3 Multi Task Learning

Multi Task Learning (MTL) is using information from multiple reward signals in the learning process. There are two key insights. The first is that because tasks have some measure of similarity, it should be possible to transfer knowledge of one task to another. Pre-training neural networks (such as with ImageNet [7], a

database with tens of millions of labeled images) can be seen as sequential multi-task learning [19], as can the entire field of transfer learning [23][34]. But MTL is more than just transferring knowledge to enable learning a new task, it is a means of improving performance on tasks. Each task trained on biases a policy's representations and feature space toward the space of generally useful features for tasks [30]. This reason is why training with an auxiliary task improves performance on a main task. While MTL has been many varied components and a relatively long history [4], this paper focuses on recent deep learning-related developments.

A common domain for MTL is in computer vision. There are many problems beyond classification - from motion prediction [15] to scene reconstruction [40] to semantic understanding [14] to abnormal behavior detection [5]- solutions to those problems all benefit from the knowledge contained in image classification. Indeed, using a network trained with ImageNet improves the performance of all of the aforementioned tasks. That is, the performance of a deep neural network on a separate task - even one not directly related to classification - achieves better accuracy when the network is first trained to correctly classify images from a common dataset. Roijers suggests that the additional tasks contribute to the success of the target task by introducing an inductive bias in the policy [28]. That is, the representations learned in the classification process are a useful subset of the general representations for image related problems, and biasing towards that is better than nothing. Indeed, the more separate tasks that are trained, the more the inductive bias is towards the general space of related problems, and the more likely that representation can rapidly learn new tasks with minimal data. This effect is

not unique to just deep artificial networks, and is also present with other machine learning techniques such as Gaussian processes and random decision forests [11], as well as working in a variety of other domains, including deep reinforcement learning (DRL) with Atari games [20].

Not all tasks use the same features, however, and it is possible that blindly using multi task learning will result in negative performance if the wrong task is used [10]. The challenge of MTL is to learn which features to share at what time - some tasks may benefit from sharing higher or lower level features then others. Recently proposed solutions include low supervision [32], cross stitch networks[19], and sluice networks[31].

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 that benefit from shared information, decision making with using the alignment metric switches between rewards by focusing on what information matters now. Alignment based methods are principled strategies for determining the relevance of subrewards while acting in the world.

2.4 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 temporal difference-based value table [35], or the weights in a neural network [38], or it can be transformed through some task-aware mapping [36]. Transfer learning has shown some success in multiagent domains [2, 36]. However, assumptions such as full observability of the world [2], and availability of inter-task mappings for knowledge transfer [36] 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*.

2.5 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 ensures that good policies for a modified reward function are also good for the original [21], and has been shown to work in multiagent domains. Reinforcement learning with reward shaping has shown success in hardware validations of multiagent systems [17]; 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 [25]. 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.

Chapter 3: Concept

3.1 Reward Alignment

Reward 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 of an agent given that all the actions of the other agents are fixed. That is, for a given agent, a local reward function R_{loc} is considered aligned with a global reward R_{glo} if:

$$\text{sign}\left(\frac{\partial R_{loc}(\mathbf{s},\mathbf{a})}{\partial \mathbf{a}}\right) = \text{sign}\left(\frac{\partial R_{glo}(\mathbf{s},\mathbf{a})}{\partial \mathbf{a}}\right) \tag{3.1}$$

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

Given two reward signals that 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, extending alignment to arbitrary states. Figure 1.1 visualizes this concept of alignment. In a domain with a main task of jointly observing with two agents a single point of interest, agents can also consider two other rewards - a reward

for moving closer to points of interests and a reward for moving closer to another agent. There are regions of the state space over which the reward for moving closer to points of interest increases with the main task reward, and there are regions over which moving closer to another agent increases the main task reward. Alignment is a metric that agents can use to select between the rewards.

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 ($\mathbf{S}_0$) given by Equation 3.2.

$$A_{\mathbf{S_0}}(R_{loc}; R_{glo}) = \frac{\sum_{\mathbf{s}\in\mathbf{S_0}}\sum_{\mathbf{a}}\sum_{\mathbf{a}'} u\left[(R_{loc}(\mathbf{s},\mathbf{a}) - R_{loc}(\mathbf{s},\mathbf{a}'))(R_{glo}(\mathbf{s},\mathbf{a}) - R_{glo}(\mathbf{s},\mathbf{a}'))\right]}{\sum_{\mathbf{s}\in\mathbf{S_0}}\sum_{\mathbf{a}}\sum_{\mathbf{a}'} 1} \tag{3.2}$$

In Equation 3.2, actions $\mathbf{a}$ and $\mathbf{a}'$ are valid actions from state $\mathbf{s}$, and $u[x]$ is the unit step function, which is equal to 1 if $x > 0$. This equation calculates a local alignment by measuring the change in reward by taking actions from the state $\mathbf{s}$. If an agent takes 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 $\mathbf{a} \in$ *Action Set* 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.

3.1.1 Calculating Alignment

In domains where the action set is well defined, finite, and reasonably sized, the alignment formulation in Equation 3.2 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 3.2 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. Moreover, because a policy trained to maximize a reward will most likely move in a direction that increases that reward, alignment can be further estimated by considering only the subset of the action space (for a particular state) in which the subreward is increasing. This sampling process is defined in Algorithm 1.

3.1.1.1 Computational Complexity of Alignment Calculation

The complexity of the calculation of a single reward depends on the particular reward. In this thesis, rewards scaled linearly and quadratically with the number of agents, but that is not a property of calculating alignment. The process of calculating alignment scales with the number of rewards.

3.1.2 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 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 in $O(\log n)$ time with the 8-vector agent state as a key. The process for generating the tree is described in algorithm 2. The alignments are calculated according to Algorithm 1. Note that the alignment for each sample is an estimation, with only the most aligned reward being remembered. Each sample is associated with only one alignment, with better (strictly positive changes in the target reward) overriding previous alignments. The alignment for the state is the reward from the most strongly aligned sample. This one-to-one mapping within the data structure is why the algorithm uses an *upsert* operation instead of an *insert*. This method makes a simplifying assumption that the subrewards have been chosen such if there is alignment in the action space, the subreward policy will most likely select from that portion of the action

Algorithm 1 Alignment Calculation

```
1: function CALC_ALIGNMENT(initW, objG, objs)
2:     agentState ← GET_STATE(initW)
3:     newW, dir ← PERTURB_WORLD(initW)
4:     dG ← EVAL(newW, objG) - EVAL(initW, objG)
5:     Alignment_o ← ∅
6:     for obj ∈ objs do
7:         dO ← EVAL(newW, obj) - EVAL(initW, obj)
8:         Alignment_o ← max (Alignment_o, ALIGN(dG, dO, Π_o))
9:     end for
   return Alignment_o, agentState, dir
10: end function
                                  ▷ Enables max operator in CALC_ALIGNMENT.
11: function GREATER_THAN_ALIGNMENT(alignA, alignB)
12:     if alignA.dG > 0 and alignA.dO > 0 then
13:         if alignB.dG > 0 and alignB.dO > 0 then
14:             return alignA.dG > alignB.dG
15:         else
16:             return True
17:         end if
18:     else
19:         if alignB.dG > 0 and alignB.dO > 0 then
20:             return False
21:         else
22:             return alignA.dO > alignB.dO
23:         end if
24:     end if
25: end function
```

space.

Algorithm 2 Alignment K-d Tree Generation

```
1: function ADD_ALIGNMENT(tree, objA, objs)
2:     for i ∈ Num Samples do
3:         alignment, state, direction ← CALC_ALIGNMENT(objA, objs)
4:         UPSERT(tree, state, (alignment.Π, direction))
5:     end for
6: end function
```

3.2 Using Alignment for Decision Making

This work introduces three different approaches to using reward alignment for decision making: selecting the action in which a reward had the highest alignment (alignment action selection), following the policy trained with the reward with the highest alignment (alignment policy selection), and learning while also selecting actions in which a reward had the highest alignment (alignment based learning). Following the policy trained with the reward with the highest alignment uses the alignment of all subrewards in a particular state across a sample set of actions to determine the most aligned subreward, and then acts according to a policy designed to maximize that subreward. Selecting the action in which the reward has the highest alignment also considers the alignment of all subrewards for the same particular state and sample set of actions, but instead selects the action that resulted in maximum alignment for one of the subrewards. Alignment based learning is a variant of transfer learning that generalizes from selecting the action that is most aligned by training a policy in tandem with alignment action selection. Agents initially take only aligned actions, but gradually use their own policies to select actions instead.

3.2.1 Following a policy of the most aligned reward

For alignment based policy selection (Algorithm 3), each agent attempts to determine the reward that is most aligned to the global reward (in a given direction), and then use the policy trained with that reward (hopefully, but not necessarily) moving in the direction of maximum alignment. An agent does so by comparing its current state to the most similar state it has seen before (or sampled randomly during an initialization period), and moving in the direction determined by the policy corresponding to the most aligned reward for that similar state.

Algorithm 3 Alignment-based policy selection

```
1: function POLICY_SELECT(tree, input_state)
2:     alignment, direction ← NEAREST_NEIGHBOR(tree, input_state)
3:     return alignment.Π(input_state)
4: end function
```

3.2.2 Selecting actions that are maximally aligned

For alignment based action selection (Algorithm 4), each agent attempts to move in the direction that has the greatest alignment between any reward and the global reward. It does so by comparing its current state to the most similar state it has seen before, and moving in the direction that had the most aligned reward for that similar state.

A difference between agent-based action selection and agent-based policy selection is that agent-based action selection is unaffected by the simplifying assumption

made in the alignment calculation that policies should move in the direction of optimal alignment. Differences in the performance of the two methods are directly attributable to the breakdown of this assumption.

Currently, the action selection method only samples over an action space that corresponds to a single time step. In more challenging domains, a roll out of multiple steps will be needed to avoid local minimums that a trained policy might be able to climb out of.

The distinction between the two methods is subtle, but important. Alignment policy selection estimates which policy is most likely to be aligned when followed, and follows it (Algorithm 3). Alignment action selection moves in the direction that is most aligned (Algorithm 4).

Algorithm 4 Alignment-based action selection

```
1: function ACTION_SELECT(tree, input_state)
2:     alignment, direction ← NEAREST_NEIGHBOR(tree, input_state)
3:     return direction
4: end function
```

3.2.3 Selecting aligned actions with learning

We combine selecting the action that the most aligned reward is most aligned in with learning to improve the performance of the selection algorithm and to make it more generalizable. Agents probabilistically select between alignment based action selection or their own neural network policy. The probability they select their own network increases each generation, and the networks are trained with a

co-operative coevolutionary algorithm (CCEA) in an identical manner to the other trained policies.

A direct transfer of the information from alignment to use as a baseline for learning is not possible because there is no neural network policy involved in the alignment based selection. Instead, agents use the alignment algorithm (either policy or action selection) like a set of training wheels, gradually decreasing their dependence on the algorithm as they train their own neural network policies. In this setup, alignment functions as an exploratory aid in the evolutionary algorithm, biasing the agents towards exploring aligned regions of the state space.

Chapter 4: Rover Domain

All experiments are performed in the simulated multi-rover exploration domain that aims to represent a team of rovers observing and exploring points of interest over an environment.

4.1 Overview

The rover domain contains a set of rovers (agents) and points of interests (POIs) in a two-dimensional continuous plane. The objective is for agents to observe the POIs after a discrete number of time steps. Each POI has an observation radius. Each agent's view is divided into four quadrants (north-east, north-west, south-east, and south-west) relative to its current heading, see Figure 4.1. The agent state (as in, the input to the agent's neural network policy) is determined by 2 sensors (POI, agent) in each quadrant which returns the density of observable POIs and and other agents in that quadrant. The density value is the sum of the values of each of the POIs or agents scaled inversely by the euclidean distance from the sensor. The input to the neural networks is therefore an 8 dimensional vector.

Thus at each time step, each agent receives a feature vector of length 8 that summarizes the world from the agent's point of view (ie, the quadrant summaries are in the agent's coordinate frame because all measurements are taken from the

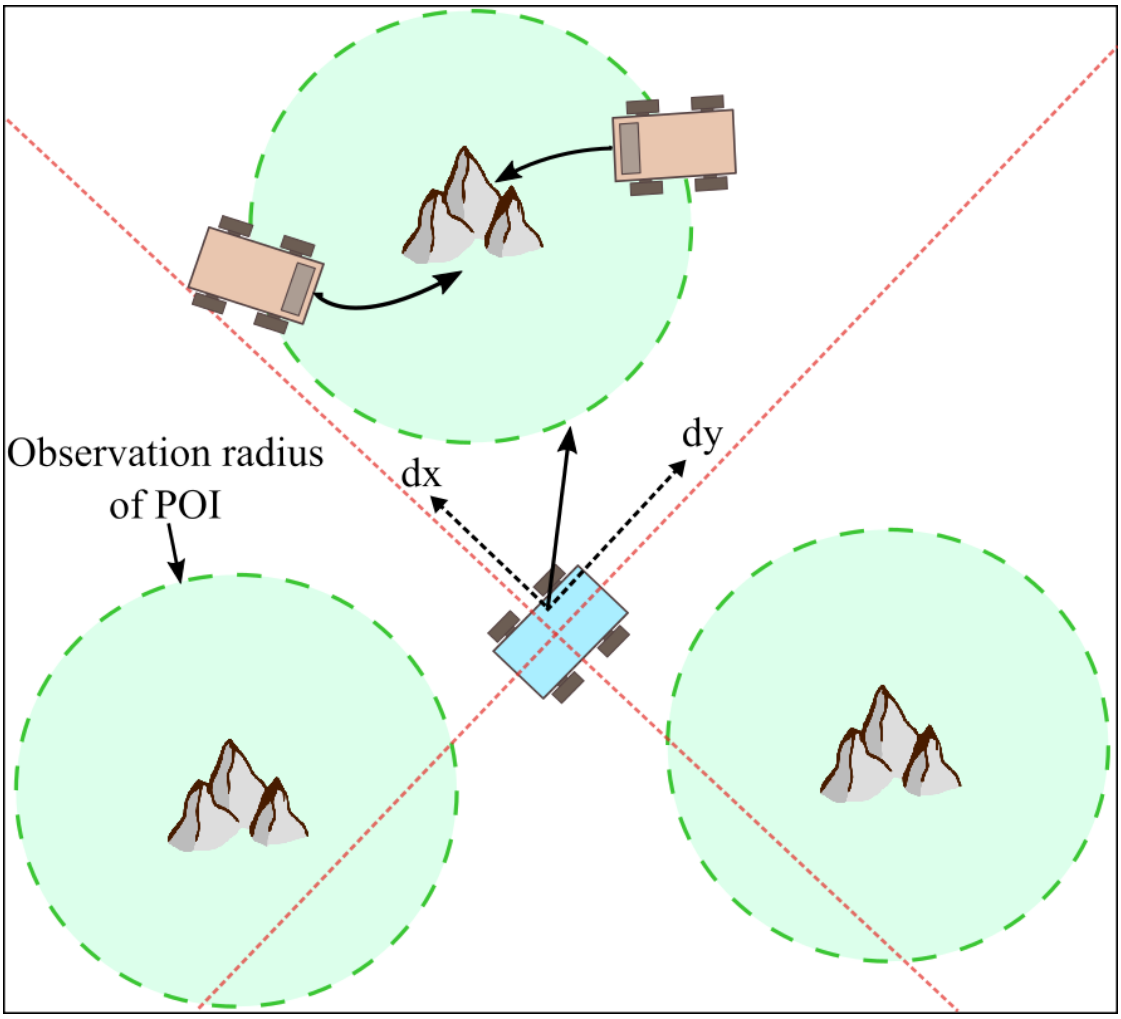

Figure 4.1: Diagram of the rover domain. The world is broken up into four quadrants relative to the rover position and orientation. POIs and fellow robots that are observed in each quadrant are summed resulting in 8 state variables. At each timestep, the robot's neural network controller yields two continuous 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.

position of the agent and with respect to the agent's heading). The agents select their actions simultaneously during the episodes using a feedforward neural network policy that takes 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 agents' trajectories at the end of the episode. During training, this reward is used as a fitness signal in the evolutionary algorithms training the agent policies.

4.2 Single Observation Required

In the simplest case, each POI must only be observed by a single agent in a given time step within its observation radius to count as observed. The reward at the end of the episode for the system is based on the closest observation over the entire episode. Each POI's maximum total reward is determined by randomly sampling a uniform distribution between 4 and 10. The equation below (4.1) describes how to calculate the reward for a domain in which POIs only require a single observation.

$$P(\mathbf{z}) = \sum_{p} \sum_{i} \frac{V_p N^1_{p,i}}{\delta_{p,i}} \tag{4.1}$$

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 agent (rounded up to one if the distance would be less than one), and $N^a_{p,i}$ is an indicator variable that is only one if agent i was the a-th closest observation to POI p and the observation was within the maximum allowable observation distance (r_{obs}), described by the

equation below.

$$N^1_{p,1} = 1 \text{ if } \delta_{p,i} < r_{\text{obs}} \text{ and } \delta_{p,i} < \delta_{p,a} \; \forall a \neq i$$

This reward can be calculated in $O(PA)$ time, with P being the number of POIs and A being the number of agents.

4.3 Multiple Observations Required

To increase the difficulty of the problem, we can increase the number of simultaneous observations required. We refer to the number of simultaneous observations as the coupling requirement, or coupling. As the coupling requirement increases, agents must learn to form teams of appropriate size.

The results for increasing coupling requirements are shown in Figure 4.2. Trials of 25 time steps were run in randomly initialized test worlds of size 30 by 30 containing 10 agents and 10 POIs, with the requirement that all observations must be within 4 units. The reward decreases as the coupling requirement increases because the problem becomes more difficult, and a satisfactory policy becomes harder to learn. The difference reward does better for a coupling requirement of 2, with no statistically significant differences for other levels of coupling. The difference reward does not learn in these higher coupling domains because the problem is not credit assignment, but state discovery: as the number of required observing agents increases, the probability of initially being in a state satisfying

those requirements decreases. If the team itself cannot receive a reward signal, an individual agent's contribution to that failure will not be helpful.

Based on the decreasing performance of CCEA with both the global reward and the difference reward after two observations, we chose to use a reward with a coupling requirement of 3 for our experiments (unless otherwise stated), meaning that every POI must have 3 agents simultaneously observe it to receive a reward. Our algorithms were effective for higher coupling requirements (we tested up through 6 simultaneous observations required), but 3 required is the last domain in which random actions and other methods can reliably get nonzero rewards. This feature makes the task difficult because of how unlikely it is that multiple agents will stumble upon the proper configuration - the reward space is very sparse with respect to the joint state-action space. Even worse from the learning perspective, is that random exploration of the joint state-action space biases away from the region of reward (the more agents there are, the less likely they are moving in the same direction).

These features make the problem an ideal one to test alignment guided exploration because the reward from just the global reward is insufficient to learn, as seen in figure 4.2 where an increasing coupling requirement results in dramatically lower reward. Because of this lack of information in just the global reward, additional rewards must add the information for the alignment based method (or any method) to succeed in compensating for the relatively nonexistent gradient for the global reward in the bulk of the state space.

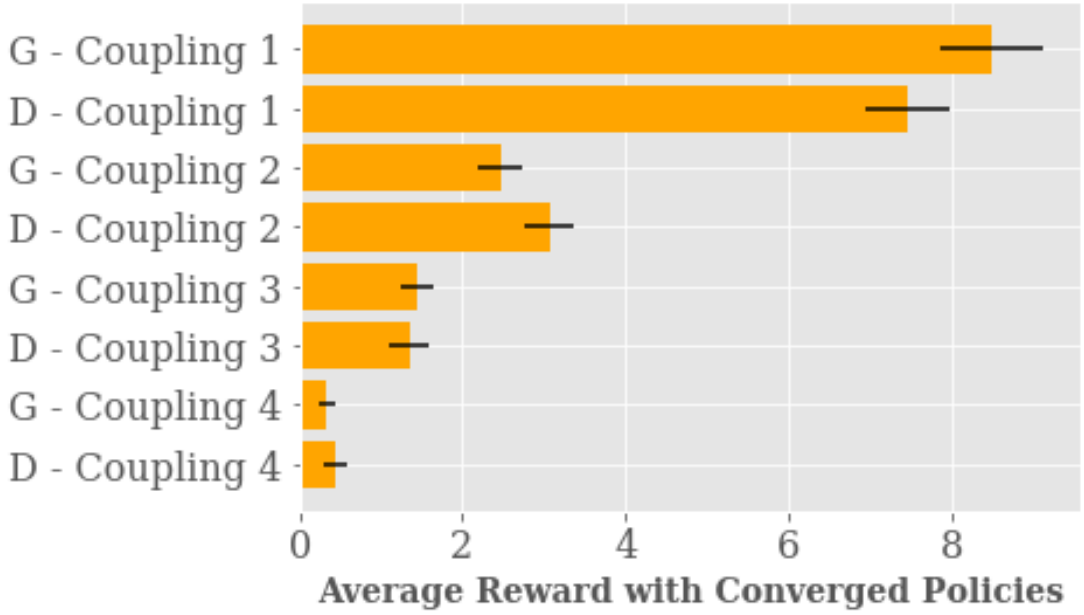

Figure 4.2: The performance of rovers in the multiagent domain decreases as the number of agents required to make a successful observation increase. Reward shaping with difference rewards (D) trends in the same direction.

The global reward, for a coupling of 3, is:

$$P(\mathbf{z}) = \sum_{p} \sum_{i} \sum_{j} \sum_{k} \frac{V_p N^1_{p,i} N^2_{p,j} N^3_{p,k}}{\frac{1}{3}(\delta_{p,i} + \delta_{p,j} + \delta_{p,k})} \tag{4.2}$$

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.

$$N^1_{p,i} = 1 \text{ if } \delta_{p,i} < \delta_0 \text{ and } \delta_{p,i} < \delta_{p,l} \; \forall l \neq i \tag{4.3}$$

$$N^2_{p,j} = 1 \text{ if } \delta_{p,j} < \delta_0 \text{ and } \delta_{p,j} < \delta_{p,l} \; \forall l \neq i, j \tag{4.4}$$

$$N^3_{p,k} = 1 \text{ if } \delta_{p,k} < \delta_0 \text{ and } \delta_{p,k} < \delta_{p,l} \; \forall l \neq i, j, k \tag{4.5}$$

This reward can be calculated in $O(PA \log k)$ time and $O(k)$ space, with P being the number of POIs, A being the number of agents, and k being the coupling requirement.

Chapter 5: Experiments

We present experiments for the three different types of alignment algorithms: alignment policy selection, alignment action selection, and alignment based learning. We propose experiments to show that no single reward or subpolicy is capable of individually outperforming alignment based methods. Additionally, we compare against the state of the art in multiagent domains - evolutionarily learning strategies with and without reward shaping.

5.1 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 (Single POI, Exclusive Single POI) 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 5.1. 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.

The rovers receive no information regarding the task prior to learning, and are trained with CCEA [24], a multiagent evolutionary algorithm described in Algorithm 5. In CCEA, each agent maintains an internal pool of potential neural networks solutions. During each episode, one network from each agent's pool is randomly selected to be a member of the team. That team is then tested. The reward the network receives is determined by the reward the team of agents receives. A generation consists of evaluating all networks within the agents pools once. After each generation, each agent eliminates bad solutions from its pool of neural networks and generates new candidates from the good solutions. We added Gaussian noise with zero mean and unit variance to a randomly selected 10% of our neural network weights as our mutation operator.

Algorithm 5 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 → 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
```

5.1.1 Single POI

Single POI are rewards agents receive for traveling within the observation radius of a POI.

$$R_{single} = \sum_{a \in A} \sum_{p \in P} \frac{V_p}{\delta_{a,p}} \tag{5.1}$$

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.

There are two Single POI objectives used: Exclusive POI in which only the closest agent scores (similar to the global objective) and Single POI in which every agent scores. They are both, however, trained on the same domain.

The computational complexity for both of these rewards is $O(AP)$.

5.1.2 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.

$$R_{shared} = \sum_{a \in A} \sum_{i} \sum_{j} \frac{N^1_{a,i} N^2_{a,j}}{\frac{1}{2}(\delta_{a,i} + \delta_{a,j})} \tag{5.2}$$

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.

Similar to the global reward, the computational complexity for these rewards is $O(A^2 \log k)$, with k being the size of the team.

5.1.3 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.

Agents are ranked according to the average distances to the closest agents that would be in the team centered on that agent. Agents are selected for teams based on descending score. Below is the equation for the score (S) of teams of size 3. The score is the individual component of R_{shared}.

$$\mathrm{S}_a = \sum_i \sum_j \frac{N^1_{a,i} N^2_{a,j}}{\frac{1}{2}(\delta_{a,i} + \delta_{a,j})} \tag{5.3}$$

These agents are, if not selected to be a member of another agents's team,

Reward	Coupling	Obs 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 5.1: 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. Single POI and Exclusive Single POI were trained identically. All experiments were trained over 1000 generations with population of 40 neural networks per agent.

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.

$$R_{exclusive} = \sum_{a \in A} \sum_{i} \sum_{j} \frac{M^1_{a,i} M^2_{a,j}}{\frac{1}{2}(\delta_{a,i} + \delta_{a,j})} \tag{5.4}$$

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.

$$M^i_{a,k} = 1 \text{ if } \forall a_i \in A \mid \mathrm{S}_{a_i} < \mathrm{LS}_a, M^i_{a_i,k} = 0 \tag{5.5}$$

The computational complexity for these rewards is $O(A^2)$. The computation is dominated by the pairwise distance calculation between all agents and subsequent per agent sum of those distances.

5.2 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 1.1). 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.)

5.3 Multiagent Rover Performance with Alignment Decision Making

Next, we test alignment, sub-policies, and other learning methods in the general rover domain with the global reward. Each POI requires 3 simultaneous agents

to observe it within a radius of 4 units to be counted as observed. We tested on several world sizes and configurations to show generalizable performance of alignment agents. For all experimental setups, the world was divided into a 3 by 3 grid, and agents were initialized only in the middle section with POIs scattered among the other eight sections. This setup aims to minimize the reward agents would receive no reward if the team remained stationary.

For the teams using alignment to select policies, sub-policies were trained on random worlds with single sub-rewards. These rewards and the domain they are training in are summarized in Table 5.1. The sub-policies were trained in simpler worlds wiht fewer agents and POIs. Additionally, the sub-policies only look at a subset of the agent input state. The Single POI objectives only look at POI quadrant summaries, and Teaming objectives only look at agent quadrant summaries.

For the first domain, 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.

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 POIs. A larger world is more difficult because it is less likely that a series of random walks from the agents will observe the POIs, thus increasing the importance of learning. The size of larger domains was chosen because it is twice the size of the smaller one. There were three domain configurations labeled "More Agents," "More POIs,"

and “Large.” They have 30 agents and 10 POIs, 10 agents and 30 POIs, and 30 agents and 30 POIs respectively. The intent behind this experimental setup was to preserve the ratio of agents to POIs of the original problem in ”More Agents” and explore the effects of increasing problem difficulty by lowering the ratio of agents to POIs. This ratio corresponds roughly to the difficulty of the problem, because the lower the ratio, the greater the need for the agents to form teams.

For alignment-based policy, 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. For alignment-based selection, at each timestep, an agent will move in the direction in which a sub-reward is most aligned.

We compare alignment 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). This reward is identical to the Single Exclusive POI subreward. 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 reward, 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.

5.4 Alignment Robustness

Alignment is a measure between different types of rewards, but it may not be always possible to correctly determine alignment. We examine the robustness of alignment to both incorporation of unhelpful objectives and the addition of deliberate noise to the alignment-based selection.

First, we examine the average rate of selection across the domains for the different rewards. Then, we compare this addition of noise to removing that reward from consideration.

5.5 Learning from Alignment

We then trained policies with alignment based learning, combining CCEA with alignment action selection. Learning is done by training a neural network policy to control actions (in a similar manner to all other neural network control policies described in this work), but incorporating information from alignment during the learning process. The agent will choose between using it's policy or using alignment to select an action, with the neural networks being evaluated after each episode normally. Agents probabilistically select between alignment based action selection or their own neural network policy. The probability they select their own network

increases each generation, and the networks are trained with CCEA in an identical manner to the other trained policies. Alignment functions in similar manner to training wheels - the novice cyclist is given a reduced state space over which to learn, with the support gradually receding as the agent learns.

A direct transfer of the information from alignment to use as a baseline for learning is not trivial because there is no neural network policy involved in the alignment based selection. The alignment map could also be learned by a supervised learning technique mapping state to action, with that trained classifier being the initial policy for the reinforcement learner.

We test alignment-based learning across all domains mentioned in this work (regular, large, more POIs, and more agents).

Chapter 6: Results

We show converged performance comparisons as there is no comparable learning-per-epoch curve to directly compare reward alignment with other techniques such as reinforcement learning. We test the general performance of agents using alignment decision making in a tightly coupled rover domain instance. The agents selecting policies based on alignment outperform agents trained on G, D, or transfer learning, as well as each of the subrewards. The agents directly going in the direction of maximal alignment achieve the highest performance. This effect is seen in both the smaller, 18 agent, 25×25 world in Figure 6.1, and in the larger 30 agent, 50×50 world with both 10 POIs in Figure 6.2 and 30 POIs in Figure 6.4.

For the 50 x 50 world with 30 agents and 10 POIs, shown in Figure 6.2, the performance of alignment policy selection agents continues to hold strong against the baseline methods. Alignment-based action selection remains the clear winner. 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.

However, in a similarly sized large world with 10 agents and 30 POIs, alignment based policy selection does not perform better than any other objective (or

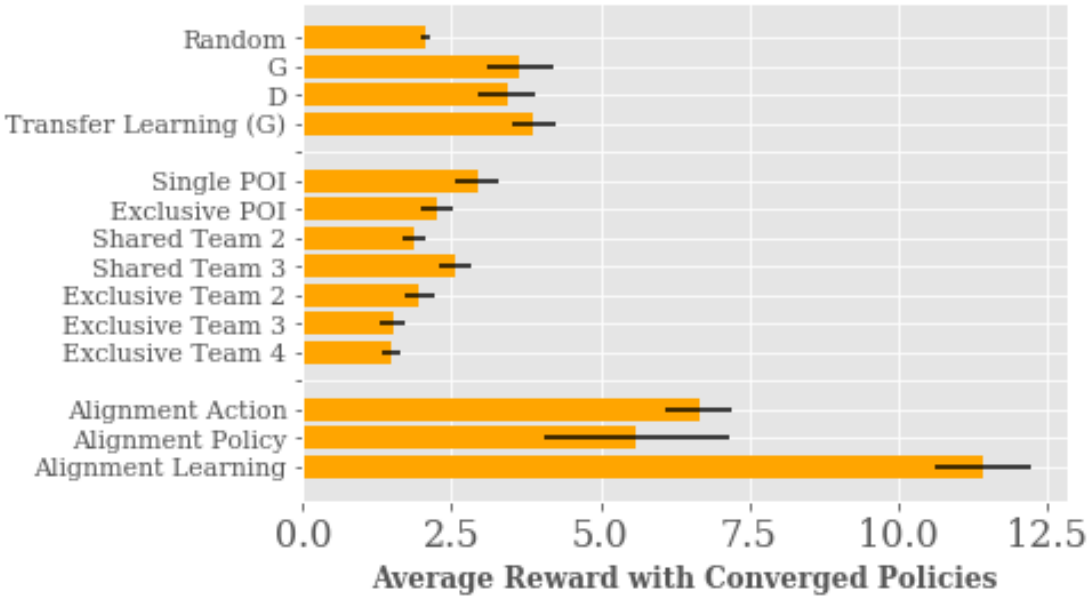

Figure 6.1: Basic world - 25x25 with 18 agents and 6 POIs. Results show converged performance for different strategies in the rover domain with observations requiring three agents. In this figure the best performance is achieved by teams using alignment action selection with learning, followed by selection without learning, and finally alignment policy selection. Due to the high number of required observations, teams trained with 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.

random), see Figure 6.3. The direct direction selection, however, still scores higher than all other policies.

In large world with 30 POIs and 30 agents, shown in Figure 6.4, the performance of the alignment action selection with and without learning is again higher than other methods. However, in this setup, the control methods of learning with the global reward and difference reward manages to match the alignment policy selection performance.

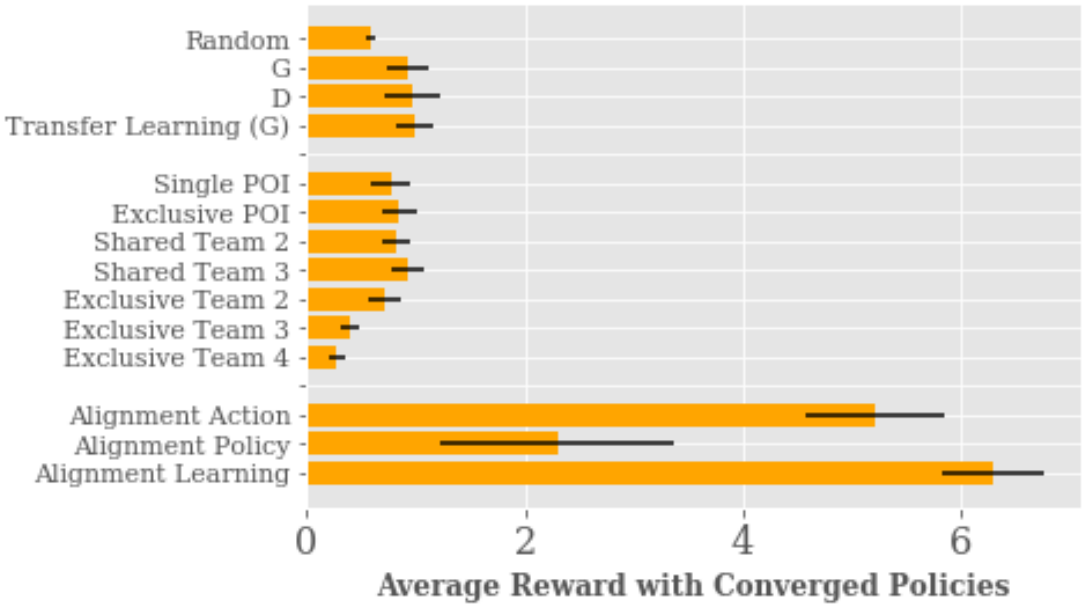

Figure 6.2: More agent world - 50x50 with 30 agents and 10 POIs. Results show converged performance for different strategies in the tightly coupled rover domain.

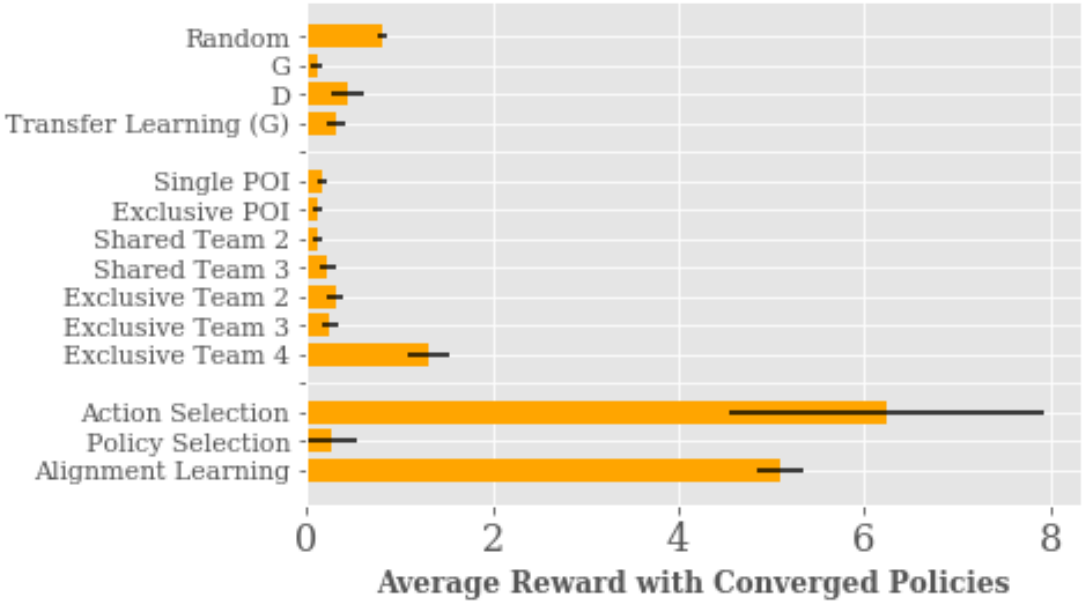

Figure 6.3: More POI world - 50x50 with 10 agents and 30 POIs. Results show converged performance for different strategies in the tightly coupled rover domain. As the number of agents decreases, the importance of both forming teams and observing POIs successfully increases because it becomes much less probably that accidental teams are formed. For this domain setup, alignment based policy selection fails to outperform single objective teaming rewards. However, alignment based action selection still dominates the other methods.

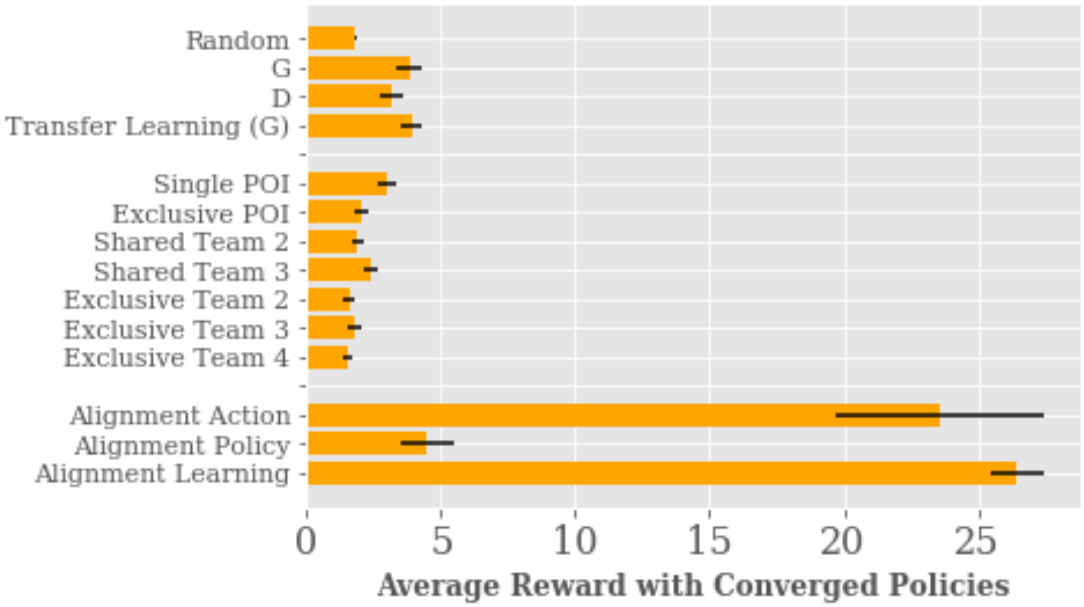

Figure 6.4: Large world - 50x50 with 30 agents and 30 POIs. Results show converged performance for different strategies in the tightly coupled rover domain. For this larger world, the global reward is able to perform better than many of the simple rewards, and within the margin of error of alignment based policy selection. However, alignment based action selection with and without learning still dominates the other methods.

6.1 Reward Alignment Case Study 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 task is captured by the most aligned reward signal, which we see in Figure 6.5. 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 chose 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 vectors 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

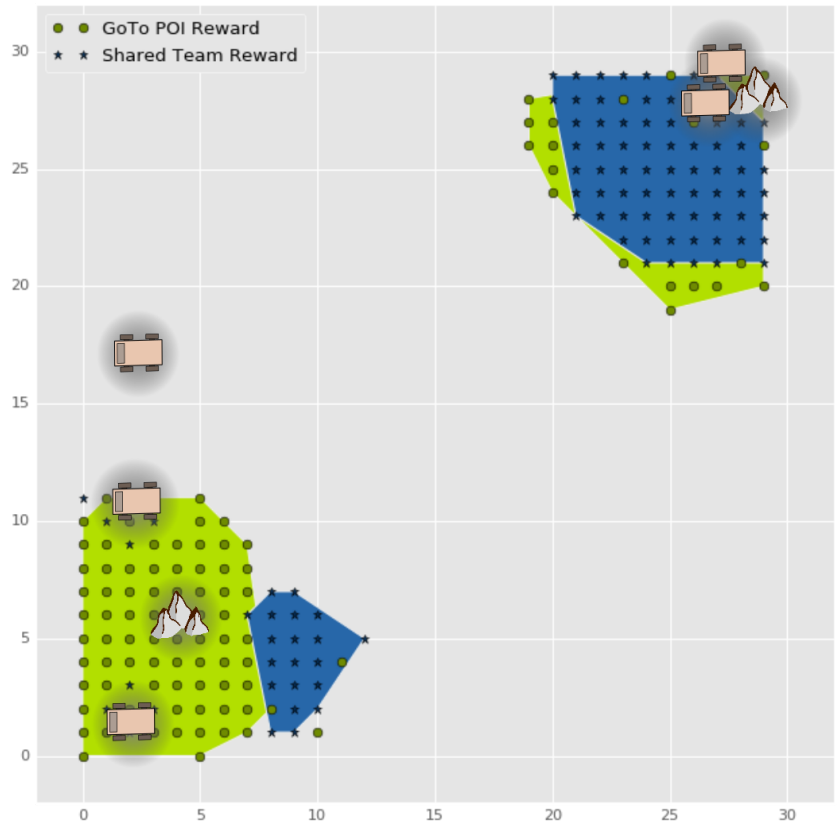

Figure 6.5: The most aligned reward for an agent placed at various points in a world. Plotted are the local reward gradients for the POI reward (blue stars) and the rover reward (green dots). The vectors shown are the most aligned reward at these points in the world, where the reward gradient and the gradient of G strongly match.

signal being calculated as most aligned in the presence of multiple POI.

In Figure 6.5, 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.

6.2 Reward Alignment Robustness to Reward Selection

We show that alignment is moderately robust to reward choice and noise. We repeatedly recorded the frequency with which a reward was most aligned with the global reward. The most aligned reward is the reward that the agent uses to make its decision, so this frequency is a measure of the importance of the reward to the decision making process. We then removed the rewards most important with that metric.

As can be see in Table 6.1, the most frequently aligned reward with the global reward with all rewards initially present is Exclusive Single POI.

Objective	Frequency (%)
Single POI	0
Exclusive POI	85
Shared Team 2	0.2
Shared Team 3	2.6
Exclusive Team 2	4.8
Exclusive Team 3	3.8
Exclusive Team 4	3.6

Table 6.1: Frequency an individual sub reward was selected as the most aligned. The majority of the time, the Exclusive POI reward is most aligned. These data were collected in a 25×25 world with 18 agents and 6 POIs over a period of 25 time steps. The frequencies are over 5400 actions.

We then repeated the experiments above without objective Exclusive Single POI. Table 6.2 shows the new reward frequencies, and Figure 6.6 shows the impact on alignment selection before and after the removal of the reward. The direct selection of direction with alignment suffers a dramatic hit to performance with

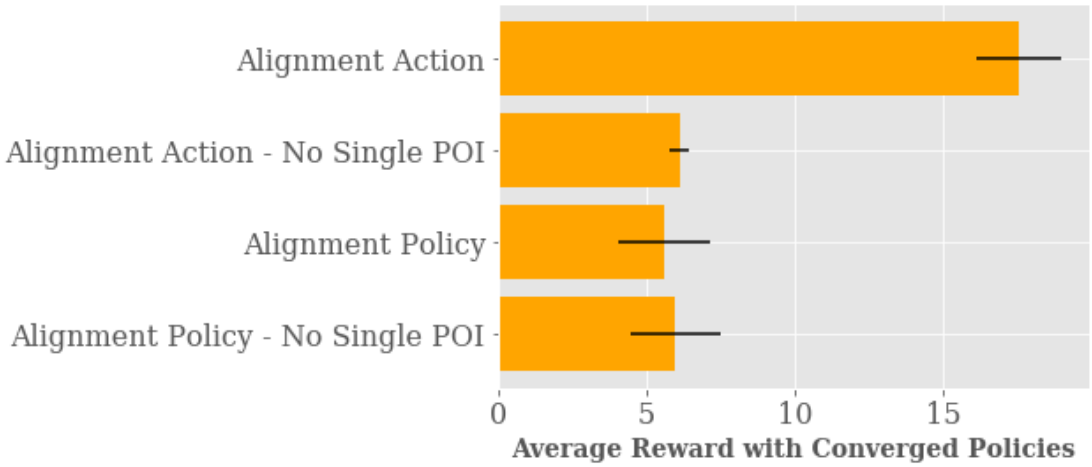

Figure 6.6: The performance of alignment based selection methods with and without the Exclusive Single POI objective (the most commonly selected objective). Alignment selection of direction achieved a substantially lower score, but alignment selection of policy did not have a statistically significant change.

the removal of the most frequently selected reward, but the selection of policy is not impacted. This result is probably because the exclusive team policies encourage dispersion, the net effect of which is similar to exploration in the direction of a POI. However, this dispersion does not improve the value of the reward at any given step, so the directional alignment fails to discover it.

To further examine the contributions of individual rewards, the above experiments were repeated with

1. All exclusive team forming rewards removed (frequencies in Table 6.3)

2. Just exclusive team 4 removed (frequencies in Table 6.4)

Objective	Frequency (%)
Single POI	0.1
Shared Team 2	13.8
Shared Team 3	14.6
Exclusive Team 2	19.3
Exclusive Team 3	27.4
Exclusive Team 4	24.8

Table 6.2: Frequency an individual sub reward was selected as the most aligned with Exclusive Single POI removed. Without that reward, the frequencies are much more evenly distributed, with the exception of Single POI. These data were collected in a 25×25 world with 18 agents and 6 POIs over a period of 25 time steps. The frequencies are over 5400 actions.

The results of alignment action selection and alignment policy selection are shown in Figure 6.7.

Alignment policy selection is robust to losing exclusive team forming 4, but not to losing all of the exclusive team forming rewards. This result is consistent with the hypothesis that having exclusive team forming rewards encourages dispersion.

Alignment action selection increases performance as exclusive team forming rewards are removed.

Objective	Frequency (%)
Single POI	0.1
Shared Team 2	48
Shared Team 3	51

Table 6.3: Frequency an individual sub reward was selected as the most aligned with Exclusive Single POI and all Exclusive Team Forming rewards removed. These data were collected in a 25×25 world with 18 agents and 6 POIs over a period of 25 time steps. The frequencies are over 54000 actions.

Objective	Frequency (%)
Single POI	0.1
Shared Team 2	24.7
Shared Team 3	21.5
Exclusive Team 2	25.2
Exclusive Team 3	28.4

Table 6.4: Frequency an individual sub reward was selected as the most aligned with Exclusive Single POI and Exclusive Team Forming 4 rewards removed. These data were collected in a 25×25 world with 18 agents and 6 POIs over a period of 25 time steps. The frequencies are over 54000 actions.

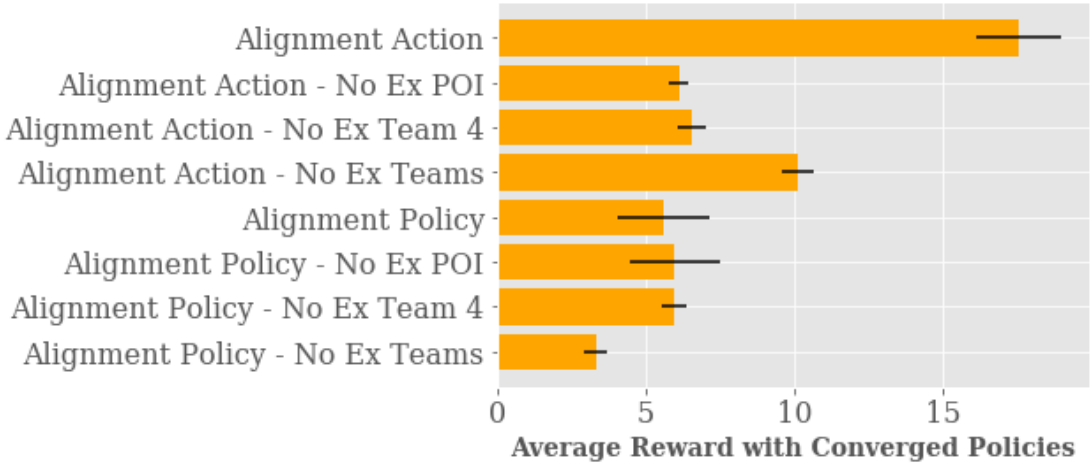

Figure 6.7: The performance of alignment based selection methods without Exclusive Single POI, and without Exclusive Teams / Exclusive Team 4 (in addition to without Exclusive Single POI reward).

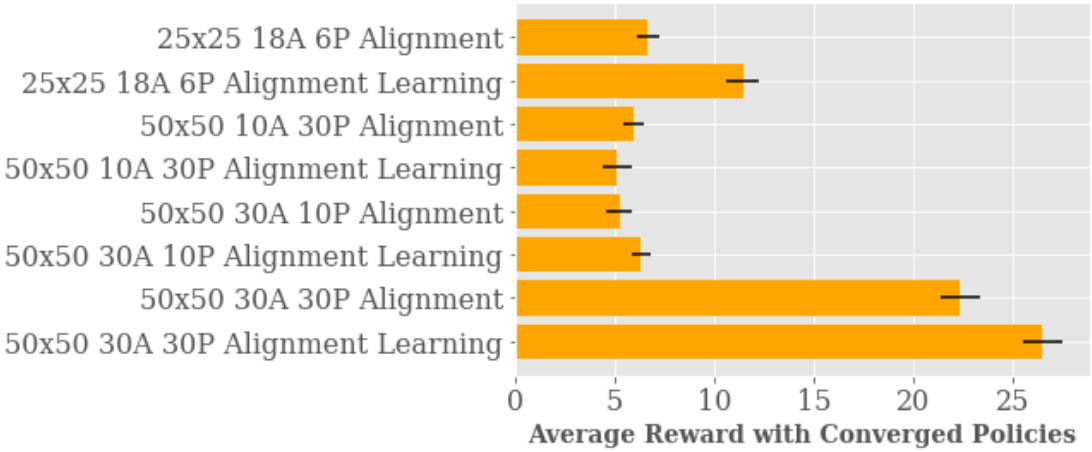

Figure 6.8: Alignment with learning. Alignment acts as training wheels for an agent training a neural network, with the neural network becoming progressively more responsible for action selection over the duration of training. Across all domains, alignment with learning is able to score equal to or greater than alignment for all domains but the 50x50 10 Agents 30 POIs.

Chapter 7: Conclusion

Alignment allowed agents to solve problems they otherwise were not able to solve using traditional methods. 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 complexity of the problem (the requirement for three simultaneous observations did not allow for the teams to stumble upon the correct action, even if they had reward shaping that would have properly attributed the credit of such an action). 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.

This work introduced three different approaches to using reward alignment for decision making, alignment policy selection, alignment action selection, and alignment based learning. Alignment policy selection uses the alignment of all subrewards in a particular state across a sample set of actions to determine the most aligned subreward, and then acts according to a policy designed to maximize that subreward. Alignment action selection considers the alignment of all subrewards for the same particular state and sample set of actions, but instead selects the action that resulted in maximum alignment for one of the subrewards. Alignment

based learning is a variant of transfer learning that generalizes from alignment action selection by training a policy in tandem with alignment action selection. Agents initially take only aligned actions, but gradually use their own policies to select actions instead.

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 6.5, the agents calculating strongly aligned rewards are able to pick which sub-rewards solve the current state best. This, in turn, provides them a structured manner to find the best action to take, which increases their overall performance.

Alignment action selection had the highest performance. As seen in Figures 6.1 and 6.2, 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. Additionally, the removal of sub-rewards decreases the performance of this approach.

Alignment policy selection allows agents to act as if they have a single policy which was trained to solve the original, complex problem. While the method did not perform as well as alignment action selection, it may be more generalizable in other domains, and it is more robust to the specifics of the subrewards. One potential 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. Moreover, alignment provides a framework to use pre-existing policies as building blocks to solve a more complex problem.

Alignment action selection, in contrast, does not require the training of any additional policies, however, it is inherently limited to a smaller region of the state space. Future work can explore alignment sampling in conjunction with some form of rollout to allow for multiple actions (especially if the action space is only movement).

Finally, alignment based learning improved upon the other two alignment algorithms by learning from a baseline performance level of alignment action selection. Alignment based learning provides an algorithm to produce a generalizable control policy more effective than the current state of the art for complex multiagent problems.

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 explanations of their actions understandable by humans 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.

Bibliography

[1] Adrian K. Agogino and Kagan Tumer. Analyzing and visualizing multiagent rewards in dynamic and stochastic domains. *Autonomous Agents and Multi-Agent Systems*, 17(2):320–338, 2008.

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

[3] Jonathan Butzke and Maxim Likhachev. Planning for multi-robot exploration with multiple objective utility functions. In *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pages 3254–3259. IEEE, 2011.

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

[5] Antonio Coronato and Luigi Gallo. Towards abnormal behavior detection of cognitive impaired people. In *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2012 IEEE International Conference on*, pages 859–864. IEEE, 2012.

[6] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and TAMT Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE transactions on evolutionary computation*, 6(2):182–197, 2002.

[7] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

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

[9] Sam Devlin, Logan Yliniemi, Daniel Kudenko, and Kagan Tumer. Potential-based difference rewards for multiagent reinforcement learning. In *Proceedings*

of the 2014 international conference on Autonomous agents and multi-agent systems, pages 165–172. International Foundation for Autonomous Agents and Multiagent Systems, 2014.

[10] Carl Doersch and Andrew Zisserman. Multi-task self-supervised visual learning. In *The IEEE International Conference on Computer Vision (ICCV)*, 2017.

[11] B Fröhlich, E Rodner, M Kemmler, and J Denzler. Large-scale gaussian process classification using random decision forests. *Pattern Recognition and Image Analysis*, 22(1):113–120, 2012.

[12] Trong Nghia Hoang, Yuchen Xiao, Kavinayan Sivakumar, Christopher Amato, and Jonathan P. How. Near-optimal adversarial policy switching for decentralized asynchronous multi-agent systems. *CoRR*, abs/1710.06525, 2017.

[13] Hisao Ishibuchi, Noritaka Tsukamoto, and Yusuke Nojima. Evolutionary many-objective optimization: A short review. In *Evolutionary Computation, 2008. CEC 2008.(IEEE World Congress on Computational Intelligence). IEEE Congress on*, pages 2419–2426. IEEE, 2008.

[14] Andrej Karpathy and Li Fei-Fei. Deep visual-semantic alignments for generating image descriptions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3128–3137, 2015.

[15] Qingshan Liu, Renlong Hang, Huihui Song, and Zhi Li. Learning multi-scale deep features for high-resolution satellite image classification. *arXiv preprint arXiv:1611.03591*, 2016.

[16] R Timothy Marler and Jasbir S Arora. Survey of multi-objective optimization methods for engineering. *Structural and multidisciplinary optimization*, 26(6):369–395, 2004.

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

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

[19] Ishan Misra, Abhinav Shrivastava, Abhinav Gupta, and Martial Hebert. Cross-stitch networks for multi-task learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3994–4003, 2016.

[20] Hossam Mossalam, Yannis M Assael, Diederik M Roijers, and Shimon Whiteson. Multi-objective deep reinforcement learning. *arXiv preprint arXiv:1610.02707*, 2016.

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

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

[23] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2010.

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

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

[26] Chen Qiao and Guo-Li Zhang. Geometric weighting method for solving multiobjective programming problems. *Journal of North China Electric Power University*, 38(6):107–110, 2011.

[27] Aida Rahmattalabi, Jen Jen Chung, Mitchell Colby, and Kagan Tumer. D++: Structural credit assignment in tightly coupled multiagent domains. *IEEE International Conference on Intelligent Robots and Systems*, 2016-Novem:4424–4429, 2016.

[28] Diederik M Roijers, Peter Vamplew, Shimon Whiteson, and Richard Dazeley. A survey of multi-objective sequential decision-making. *Journal of Artificial Intelligence Research*, 48:67–113, 2013.

[29] Diederik Marijn Roijers, Shimon Whiteson, Frans A Oliehoek, et al. Computing convex coverage sets for faster multi-objective coordination. *J. Artif. Intell. Res.(JAIR)*, 52:399–443, 2015.

[30] Sebastian Ruder. An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098*, 2017.

[31] Sebastian Ruder, Joachim Bingel, Isabelle Augenstein, and Anders Søgaard. Sluice networks: Learning what to share between loosely related tasks. *arXiv preprint arXiv:1705.08142*, 2017.

[32] Anders Søgaard and Yoav Goldberg. Deep multi-task learning with low level tasks supervised at lower layers. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 231–235, 2016.

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

[34] Matthew E. Taylor and Peter Stone. Transfer learning for reinforcement learning domains: A survey. *The Journal of Machine Learning Research*, 10:1633–1685, 2009.

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

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

[37] Logan Yliniemi and Kagan Tumer. Paccet: An objective space transformation to iteratively convexify the pareto front. In *Asia-Pacific Conference on Simulated Evolution and Learning*, pages 204–215. Springer, 2014.

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

[39] Qingfu Zhang and Hui Li. Moea/d: A multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on evolutionary computation*, 11(6):712–731, 2007.

[40] Cheng Zhao, Li Sun, and Rustam Stolkin. A fully end-to-end deep learning approach for real-time simultaneous 3d reconstruction and material recognition. In *Advanced Robotics (ICAR), 2017 18th International Conference on*, pages 75–82. IEEE, 2017.

[41] Eckart Zitzler, Marco Laumanns, and Lothar Thiele. Spea2: Improving the strength pareto evolutionary algorithm. *TIK-report*, 103, 2001.

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