Multi-Agent Reward Analysis for Learning in Noisy Domains

Adrian K. Agogino
UCSC, NASA Ames Research Center
Mailstop 269-3
Moffett Field, CA 94035
adrian@email.arc.nasa.gov

Kagan Tumer
NASA Ames Research Center
Mailstop 269-4
Moffett Field, CA 94035
ktumer@mail.arc.nasa.gov

Abstract

In many multi-agent learning problems, it is difficult to determine, a priori, the agent reward structure that will lead to good performance. This problem is particularly pronounced in continuous, noisy domains ill-suited to simple table backup schemes commonly used in TD(λ)/Q-learning. In this paper, we present a new reward evaluation method that provides a visualization of the tradeoff between coordination among the agents and the difficulty of the learning problem each agent faces. This method is independent of the learning algorithm and is only a function of the problem domain and the agents' reward structure. We then use this reward property visualization method to determine an effective reward without performing extensive simulations. We test this method in both a static and a dynamic multi-rover learning domain where the agents have continuous state spaces and where their actions are noisy (e.g., the agents' movement decisions are not always carried out properly). Our results show that in the more difficult dynamic domain, the reward efficiency visualization method provides a two order of magnitude speedup in selecting a good reward. Most importantly it allows one to quickly create and verify rewards tailored to the observational limitations of the domain.

1. Introduction

Recent advances in distributed learning methods have addressed how to best create rewards that promote coordination in a multi-agent system [7, 9, 12]. This is a fundamental challenge that applies to most multi-agent learning problems, but particularly to learning in dynamic environments. Indeed, most coordination methods that perform well in static environments often perform poorly in dynamic environments [5]. In this paper, we present a reward evaluation method that directly addresses this issue by explicitly visualizing the coordination properties of a reward in both static and dynamic environments. This reward evaluation method is based on two important properties in multi-agent coordination:

1. how well the reward promotes coordination among agents in different parts of a domain's state-space; and
2. how easy it is for an agent to learn to maximize that reward.

These reward visualization methods provide the ability to predict the reward performance in a given domain without the need for lengthy learning trials. Furthermore, this method can be used to create either new sets of coordination mechanisms or new reward structures based on the specific needs of the domain.

We explore the agent reward design and visualization in a continuous rover problem where a set of rovers learn to navigate and collect information in an unknown environment based on their noisy sensor inputs [2]. Reinforcement learning and credit assignment is particulary challenging in this case because traditional table-based reinforcement learning methods such as Q-learing, TD(λ) and Sarsa learners are ill-suited to this domain [8]. Instead, we select a direct policy search method where the full control policy is evaluated after each learning episode. Note that this domain is not only more realistic but also significantly more difficult than previous multi-rover coordination problems where agents learned to take discrete actions in a static grid-world setting [9]. Therefore, having well tailored and computationally tractable agent rewards is particularly important in this domain.

In this paper we provide evaluation and visualization methods for multi-agent coordination problems in noisy domains with continuous state spaces. We discuss three types of agent rewards that vary in how well they promote coordination and how easy it is for the agents to learn them. A new visualization method is then used to determine which reward is best suited in the Continuous Rover Problem. In addition, the visualization is used to provide new agent rewards that take the rovers' partial observation limitations into account while retaining much of the salient features (e.g., coordination) of the full reward. Section 2 describes the key reward properties required for evaluating agent rewards and discusses three types of rewards. Section 3 presents the Continuous Rover Problem, and provides the simulation details. Section 4 presents the visualization results that allow the evaluation of the rewards, and Section 5 presents the simulation results.

2. Rewards for Agent Coordination

In this work, we focus on cooperative multi-agent systems where each agent i is taking actions to maximize its own agent reward g_i, and where the performance of the full system is measured by the global reward G. The system state z is decomposed into a component that depends on the state of agent i, denoted by z_i, and a component that does not depend on the state of agent i, denoted by z_{-i}. (We will use the notation $z = z_i + z_{-i}$ to concatenate the state vectors.) Note that though agent i may or may not influence the full state z, both G and g_i are functions of z, the full state of the system.

2.1. Factoredness and Learnability

There are two properties that are crucial to producing cooperative multi-agent systems in which agents acting to optimize their own agent rewards will also optimize the provided global reward. The first, concerns "aligning" the agent rewards of the agents with the global reward. For an agent i, let us define the **degree of factoredness** (a generalization of factoredness presented in [12, 10]) between the rewards g_i and G at point z as:

$$\mathcal{F}_{g_i} = \frac{\sum_{z'} u[((g_i(z) - g_i(z'))(G(z) - G(z'))]}{\sum_{z'} 1} \tag{1}$$

where the states z and z' only differ in the states of agent i, and $u[x]$ is the unit step function, equal to 1 if $x > 0$. Intuitively, the degree of factoredness gives the percentage of states in which a change in the action of agent i has the same impact on g_i and G. A high degree of factoredness means that the agent reward g_i is aligned with the global reward G. As a trivial example, any system in which all the agent rewards equal G has a degree of factoredness of 1.

The second property measures the dependence of a reward on the actions of a particular agent as opposed to all the other agents. Let us first define the point learnability of reward g_i, between state z and z' as the ratio of the change in g_i due to a change in the states of agent i over the change in g_i due to a change in the states of other agents:

$$L(g_i, z, z') = \frac{\|g_i(z) - g_i(z - z_i + z'_i)\|}{\|g_i(z) - g_i(z' - z'_i + z_i)\|} \tag{2}$$

where z' is an alternate to state z (e.g., in the numerator of Eq 2, agent i's state is changed from z to z', whereas in the denominator, the state of all other agents is changed from z to z'). The **learnability** of a reward g_i is then given by:

$$L(g_i, z) = \frac{\sum_{z'} L(g_i, z, z')}{\sum_{z'} 1} \tag{3}$$

Intuitively, the higher the learnability, the more g_i depends on the move of agent i, i.e., the better the associated signal-to-noise ratio for i. Therefore, higher learnability means it is easier for i to receive large values of its reward. Note that both learnability and factoredness are computed local to a particular state. Later we analyze how these properties change through the state space.

2.2. Multi-Agent Rewards

The selection of a reward that provides the best performance hinges on balancing the degree of factoredness and learnability for each agent. In general, a highly factored reward will have low learnability and a highly learnable reward will have low factoredness [12]. In this work, we analyze three different rewards that provide different trade-offs between learnability and factoredness: T_i, the team game reward (Eq. 4), P_i, the perfectly learnable reward (Eq. 5) and D_i, the difference reward (Eq. 6) given by:

$$T_i \equiv G(z) \tag{4}$$
$$P_i \equiv G(z_i) \tag{5}$$
$$D_i \equiv G(z) - G(z_{-i}). \tag{6}$$

T_i provides the full global reward to each agent. It is fully factored by definition, but because each agent's reward depends on the states of all the other agents, it generally has poor learnability, a problem that get progressively worse as the size of the system grows. P_i provides the component of the global reward that depends on the states of agent i. Because it does not depend on the states of other agents, P_i is "perfectly learnable" having infinite learnability. However, depending on the domain, it may have a low degree of factoredness. D_i provides rewards that have high factoredness, because the second term of Eq. 6 does not depend on i's states [12]. Furthermore, D_i usually has better learnability than does T_i, because the second term of D_i removes some of the effects of other agents (i.e., noise) from i's reward. While having good properties, this reward is often impractical to compute because it requires a lot of knowledge about z to compute $G(z_{-i})$. In practice either of the three rewards may be the best choice depending on their properties in a particular domain.

3. Continuous Rover Problem

In this section, we define the "Continuous Rover Problem," that will be used illustrate the importance of visualization and proper reward selection in a difficult, noisy, continuous, multi-agent domain. In this problem, multiple rovers try to observer points of interest (POIs) on a two dimensional plane. A POI has a fixed position on the plane and has a value associated with it. The value of the information from observing a POI is inversely related to the distance the rover is from the POI. In this paper the distance metric will be the squared Euclidean norm, bounded by a minimum observation distance, d:[1]

$$\delta(x,y) = min\{\|x-y\|^2, d^2\} \ . \tag{7}$$

While any rover can observe any POI, as far as the global reward is concerned, only the closest observation counts[2]. The full system, or global reward for an episode is given by:

$$G = \sum_j \frac{V_j}{\min_i \delta(L_j, L_i)} \ , \tag{8}$$

where V_j is the value of POI j, L_j is the location of POI j and L_i is the location of rover i.

At every time step, the rovers sense the world through eight continuous sensors. From a rover's point of view, the world is divided up into four quadrants relative to the rover's orientation, with two sensors per quadrant (see Figure 1). For each quadrant, the first sensor returns a function of the POIs in the quadrant. Specifically the first sensor for quadrant q returns the sum of the values of the POIs divided by their squared distance to the rover:

$$s_{1,q,i} = \sum_{j \in I_q} \frac{V_j}{\delta(L_j, L_i)} \tag{9}$$

where I_q is the set of observable POIs in quadrant q. The second sensor returns the sum of square distances from a rover to all the other rovers in the quadrant:

$$s_{2,q,i} = \sum_{i' \in N_q} \frac{1}{\delta(L_{i'}, L_i)} \tag{10}$$

where N_q is the set of rovers in quadrant q.

3.1. Simulation Set-up

With four quadrants and two sensors per quadrant, there are a total of eight continuous inputs. This eight dimensional sensor vector constitutes the state space for a rover.

1 The square Euclidean norm is appropriate for many natural phenomenon, such as light and signal attenuation. However any other type of distance metric could also be used as required by the problem domain. The minimum distance is included to prevent singularities when a rover is very close to a POI

2 Similar rewards could also be made where there are many different levels of information gain depending on the position of the rover. For example 3-D imaging may utilize different images of the same object, taken by two different rovers.

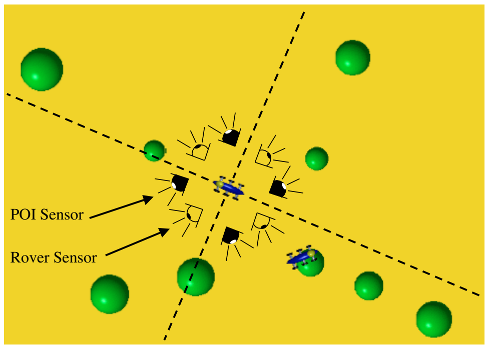

Figure 1. Diagram of a Rover's Sensor Inputs. The world is broken up into four quadrants relative to rover's position. In each quadrant one sensor senses points of interests, while the other sensor senses other rovers.

At each time step the rover uses its state to compute a two dimensional action. The action represents an x,y movement relative to the rover's location and orientation. The mapping from state to action is done with a multi-layer-perceptron (MLP), with 8 input units, 10 hidden units and 2 output units. The MLP uses a sigmoid activation function, therefore the outputs are limited to the range $(0, 1)$. The actions, dx and dy, are determined from subtracting 0.5 from the output and multiplying by the maximum distance the rover can move in one time step: $dx = d(o_1 - 0.5)$ and $dy = d(o_2 - 0.5)$ where d is the maximum distance the rover can move in one time step, o_1 is the value of the first output unit, and o_2 is the value of the second output unit. To better simulate the inaccuracies and imperfections of a rover operating in the real world, ten percent noise is added to each action. The MLP for a rover is chosen through simulated annealing, where its weights are modified and selected with preset probabilities. Note, this is a form of direct policy search, where the MLPs are the policy [3].

In these simulations, there are thirty rovers, and each episode consists of 15 time steps. The world is 100 units long and 115 units wide. All of the rovers start the episode near the center (60 units from the left boundary and 50 units from the top boundary). The maximum distance the rovers can move in one direction during a time step, d, is set to 10. The minimum distance, d, used to compute δ is equal to 5. System performance is measured by how well the rovers are able to maximize the sum of global rewards for an episode, though each rover is trying to maximize its own agent reward, discussed below.

3.2. Rover Rewards

In this paper three different types of agent rewards are tested in the Rover Problem. The first reward is the team game reward (T_i) where the agent reward is set to the global reward given in equation 8. The second reward is the "perfectly learnable" reward (P_i):

$$P_i = \sum_j \frac{V_j}{\delta(L_j, L_i)} \tag{11}$$

Note that P_i is equivalent to T_i when there is only one rover. It also has infinite learnability as defined in Section 2 (denominator is equal to zero since for P_i $g_(z' - z'_i + z_i) = g_i(z)$). However, P_i is not factored. Intuitively P_i and T_i offer opposite benefits, since T_i is by definition factored, but has poor learnability. The third reward is the difference reward. It does not have as high learnability as P_i, but is still factored like T_i . For the rover problem, the difference reward, D_i, is defined as:

$$\begin{aligned} D_i &= \sum_j \frac{V_j}{\min_{i'} \delta(L_j, L_{i'})} - \sum_j \frac{V_j}{\min_{i' \neq i} \delta(L_j, L_{i'})} \\ &= \sum_j I_{j,i}(z) \frac{V_j}{\delta(L_j, L_i)} \end{aligned}$$

where $I_{j,i}(z)$ is an indicator function, returning one if and only if POI j is the closest rover to L_j. The second term of the D_i is equal to the value of all the information collected if rover i is not in the system. Note that in practice it may be difficult to compute this reward since each rover needs to know the locations of all of the other rovers. It may even be more difficult to compute than the team game reward, T_i, since T_i is the same for all the rovers. In many cases T_i can be computed once and then broadcast to all the agents.

3.3. Static and Dynamic Environments

In the static environment, the set of POIs remained fixed for all learning episodes. The POI distributions ranged from randomly distributed across the state to checkerboard patterns of uniform POIs. The results and insights gained from visualization were qualitatively similar in all cases. To illustrate the impact of visualization, we selected the POI distribution depicted in Figure 2, which required a moderate amount of coordination. The 15 POIs to the left have value 3.0, and the lone POI to the right has of 10.0.

For the dynamic environment, the POI distribution changed every 15 time steps, and the rovers faced a different configuration at each episode. In each episode, there were one hundred POIs of equal value, distributed randomly within a 70 by 70 unit squared centered on the rovers' starting location. In the static environment, the rovers could learn specific control policies for a given configuration of POIs. This type of learning is most useful when the rovers learn on a simulated environment that closely matches the environment in which they will be deployed. However, in general it is more desirable for the

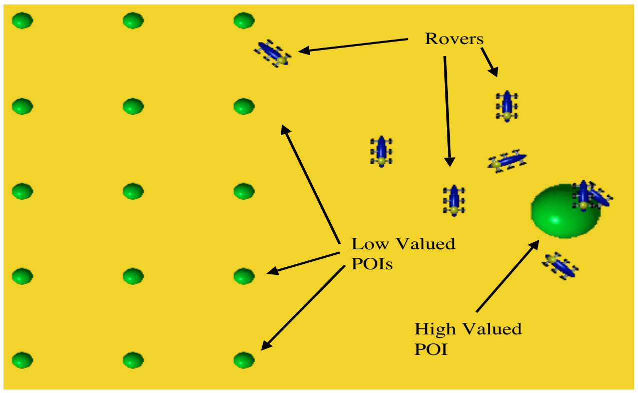

Figure 2. Diagram of Static Environment. Points of interests are at fixed locations for every episode.

rovers to directly learn the sensor/action mapping independently from the specific POI configuration, so that they can generalize to POI configurations that may be significantly different than the ones in which they were trained. The dynamic environment experiment tests the rovers' ability to generalize in constantly changing environmental conditions.

This type of problem is common in real world domains, where the rovers typically learn in a simulator and later have to apply their learning to the environment in which they are deployed. Note that this is a fundamentally difficult learning problem because: 1) the environment changes every episode, 2) noise is added to the actions of the rovers, 3) the state space is continuous, and 4) thirty rovers must coordinate. Therefore, the selection of the agent reward is critical to success and many rewards that can be used in more benign domains (e.g., grid world rovers) are unlikely to provide satisfactory results.

4. Reward Visualization

Visualization is an important part of understanding the inner workings of many systems, but particularly those of learning systems [4, 11, 1, 6]. This paper focuses on visualizing reward properties to aid in both agent reward evaluation and design. To analyze the rewards in a specific domain, we plot the learnability and factoredness of a reward measured at a set of states in the domain. This visualization helps determine which of the many possible rewards one expects to perform well in a particular domain.

The analysis starts by recording the states observed by agents taking a random set of actions [3]. For each reward,

3 States could also be recorded during learning, perhaps changing results, though preliminary analysis has shown little difference.

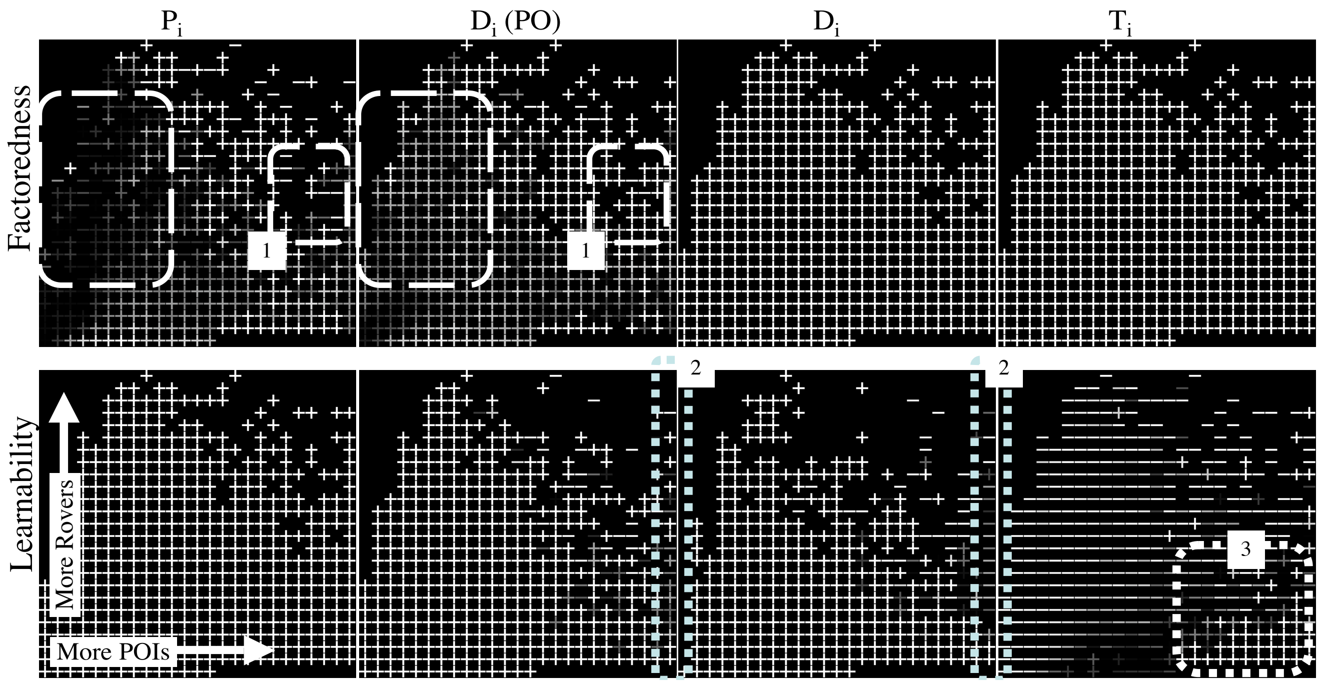

Figure 3. Factoredness and Learnability Visualization in Static Environment. First row shows factoredness of four rewards and second row shows their learnability. The visualization is a projection of an agent's state space, with increasing x values corresponding to states closer to POIs and increasing y values corresponding to states where the agent is closer to other agents. P_i has low factoredness and is anti-factored for much of region 1. D_i under partial observability ($D_i(PO)$) is much more factored. $D_i(PO)$ has higher learnability than D_i, especially in region 2. T_i generally has low learnability, but is sufficient in region 3, corresponding to regions close to POIs.

we compute the learnability and factoredness by sampling Equations 1-3. The learnability and factoredness values for each state are projected onto a two-dimensional plane, using a domain dependent projection. The projection is then broken up into fixed sized squares and all the values within a square are averaged.

In a learnability visualization, points where an agent's action influences its reward more than the actions of other agents are represented with a "+" symbol. The lighter the "+" symbol, the more an agent influences its own reward. Points where an agent's actions influence its reward less than the actions of other agents are represented with a "-" symbol. The lighter the "-" symbol, the less an agent influences its reward. In factoredness visualization, points where an agent's reward is aligned with the global reward more often than random are represented with a "+" symbol. The lighter the "+" symbol the more factored the reward is. Points where an agent's reward is aligned with the global reward less often than random (anti-aligned) are represented with a "-" symbol. The lighter the "-" symbol the more anti-factored the reward is.

In this domain, the projection axes are formed using the eight sensor values used by the rovers. The x axis of the projection corresponds to the sum of the four sensor values corresponding to POI distance, and the y axis corresponds to the sum of the four sensor values corresponding to other rover distance. Therefore values at the left side of the visualizations correspond to states where a rover is far away from the POIs, and values at the right side of the visualizations correspond to states where the rover is close to the POIs. Similarly, values at the bottom of a visualizations correspond to states where a rover is not close to any other rover, and areas towards the top of the visualizations correspond to states where the rover is close to other rovers.

4.1. Visualization in Static Environments

Figure 3 shows the learnability and factoredness visualizations for the static environment. P_i is highly factored in some parts of the state space, particularly the lower right corner. That space corresponds to conditions where there are many POIs but few other rovers in the rover's vicinity. It is not surprising that in such conditions where coordination is not relevant this reward provides the right incentives.

It is important to note that P_i has high learnability across the board, a result that is expected from how the reward is constructed. While P_i has high learnability across the board, and is therefore easy for the agents to learn. This visualization implies that in many states it results in the agent learning to take the wrong actions due to low factoredness. However, since P_i has better factoredness than random, for most states, we expect agents using P_i in this environment to reach a reasonable level of proficiency.

The situation is almost entirely reversed for T_i in this environment. It is by definition fully factored (except for states that have not been sampled, which show up as black in Figure 3), but has low learnability almost across the board. T_i has good learnability only on the right side of the visualization, corresponding to states where the rover is close to the POIs. This is an important part of the state space so we expect that agents using T_i to learn in this domain, though learning will be slow since the agents receive proper reinforcement signals only after they stumble upon regions with POIs.

D_i on the other hand is both fully factored and highly learnable. However, to compute D_i, a rover needs to be able to observe all of the other rovers which may be impractical in many domains (note, T_i also requires this). Instead we compute the partially observable D_i, where only the rovers within a radius equal to the maximum distance a rover can move in one time step are observed. This is a severe restriction that forces the agents to focus on less than 3% of the state space at any time in search of other rovers. While this reward is no longer fully factored, the factoredness visualization (labeled $D_i(PO)$) shows that the reward is still reasonably factored. In addition if we look at the right side of the learnability visualization for D_i and $D_i(PO)$ (vertical rectangle marked 2 in Figure 3), we see that $D_i(PO)$ is more learnable in this part of the state space. Considering this part of the state corresponds to the important area where a rover is close to a POI, we expect agents using $D_i(PO)$ to perform even better than agents using D_i in this static environment domain.

For the static domain, we can gain additional insight into the differences between the rewards by displaying the factoredness visualization projected directly on the x, y domain in which the rovers move (note that this projection will not usually be effective in a changing environment where important regions keep shifting). This visualization shows how the rewards map to actions directly taken by the rovers. Figure 4 shows the factoredness for $D_i(PO)$ and P_i (on this projection, T_i and D_i are fully factored, meaning each square is a light "+"). Note that around the POIs, both rewards are factored. However, there is an anti-factored boundary for P_i between the two regions. That means that agents are restricted to the right or left hand side of the x, y grid, and will not cross that boundary if doing so would benefit the global reward. This means the performance of P_i will be particularly sensitive to the initial random actions taken by the rovers. Notice that though not highly factored in that region, $D_i(PO)$ has two "bridges" to cross this region and furthermore is lightly factored rather than anti-factored in the rest of that region. This implies that $D_i(PO)$ will not have factoredness problems in this domain.

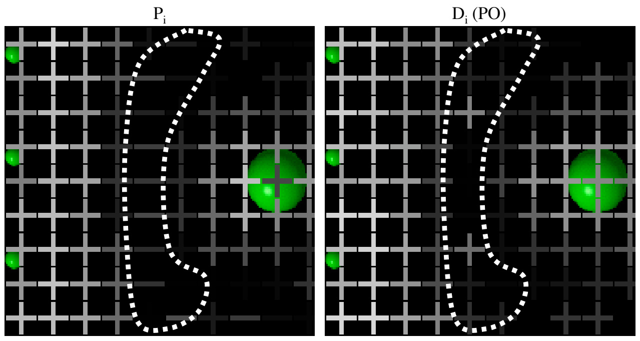

Figure 4. Factoredness Projected onto Domain Coordinates. Factoredness of P_i and $D_i(PO)$ is projected onto the x,y coordinates of the domain environment instead of onto the feature space used by rovers. "+" represents factoredness and "-" represents anti-factoredness. P_i has an anti-factored boundary preventing agents from moving from one region to the other.

4.2. Visualization in Dynamic Environments

Figure 5 shows the factoredness and learnability visualizations for dynamic environments. They show that in this more difficult environment, neither P_i nor T_i are acceptable. The factoredness deficiencies of P_i are amplified in this environment as are the learnability deficiencies of T_i. In fact the learnability is so low that there is reason to expect T_i to perform marginally better than a random algorithm. P_i is only consistently factored in the bottom left part of the visualizations, corresponding to unimportant locations where the rover is not close to any POIs or close to any other rover. In fact, in more important areas of the state space, P_i is often anti-factored, leading one to expect agents using P_i to perform very poorly in this environment.

In contrast, D_i is both highly learnable and highly factored in this domain. In fact, there is little difference between the learnability/factoredness charts of D_i in this dynamic domain and in the static domain. Given that D_i is fully factored we would expect rovers using D_i to perform very well. However, again D_i is difficult to compute in practice, as it requires a rover to know the locations of all of the other rovers. As in the static domain we can com-

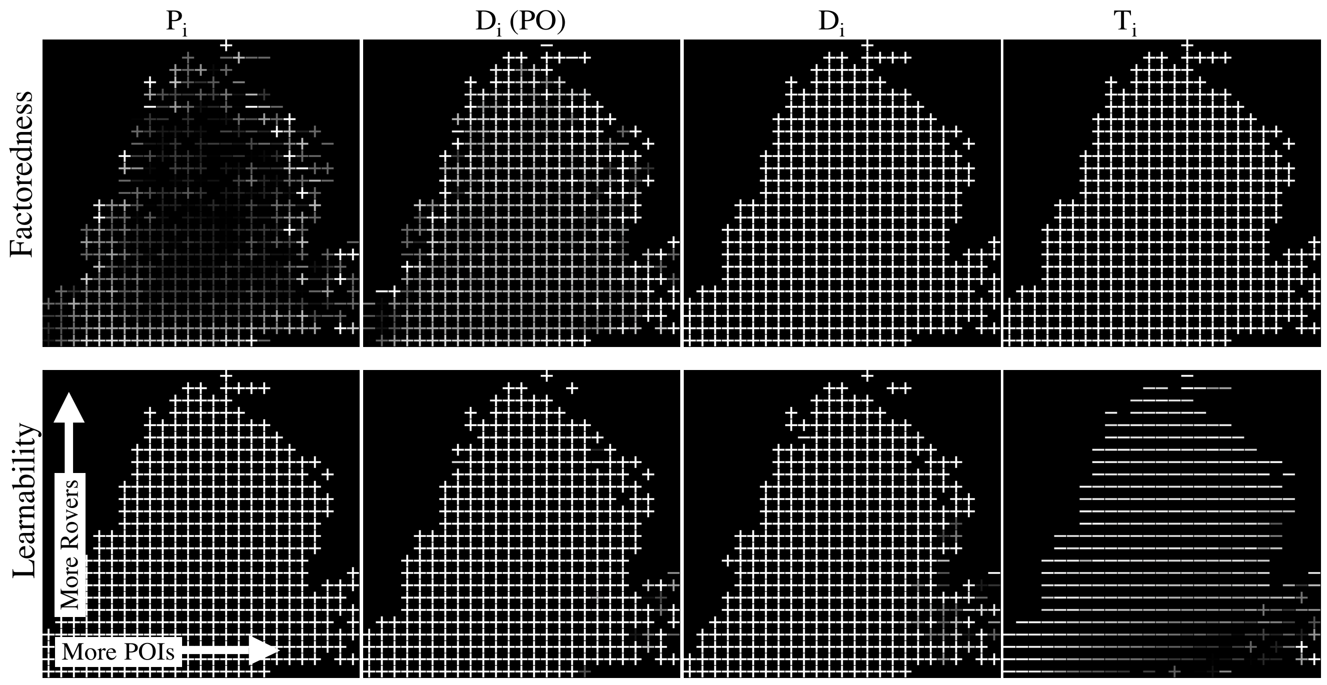

Figure 5. Factoredness and Learnability Visualization in Dynamic Environments. First row shows factoredness the four rewards and second row shows their learnability. The visualization is a projection of an agent's state space. The visualizations show that P_i has very low factoredness and T_i has very low learnability. $D_i(PO)$ (computed with partial observability) still has high factoredness.

pute $D_i(PO)$ where the rover can only observe other rovers within a radius equal to the maximum distance it can move at one time step. Though not as high as that of D_i, the factoredness of $D_i(PO)$ is still consistently high. Therefore we expect rovers using $D_i(PO)$ to significantly outperform both T_i and P_i.

5. Reward Performance

In this section we show the results from a set of experiments in both the static environment and dynamic environment to evaluate the effectiveness of the rewards in these domains. The experiments confirm the expectation obtained from the factoredness and learnability visualizations.

Figure 6 shows results from the static environment. The rovers using P_i learned quickly, but did not converge to good solutions. This is consistent with the high learnability/low factoredness properties of P_i that were apparent in the visualizations. In contrast agents using T_i were able to keep improving their performance through learning, and were able to surpass the performance of P_i. However as predicted from the learnability visualization, these rovers learn slowly, so T_i may be a poor choice of reward in quick learning is needed. As expected, rovers using D_i with full observability performed very well, since D_i is both highly learnable and fully factored. More interestingly, the rovers using $D_i(PO)$ performed even better though $D_i(PO)$ is not fully factored. This confirms that the gains in learnability more than offset the slight loss in factoredness shown in the visualizations. Note this is remarkable, since $D_i(PO)$ is in fact significantly easier to compute than D_i.

Figure 7 shows that rovers using T_i or P_i perform very poorly in the dynamic environments as predicted from the learnability and factoredness visualizations. The performance of rovers using P_i actually declines with learning, highlighting the fact that P_i leads the rovers to learn the *wrong* thing. This results confirms the intuition that highly learnable but poorly factored rewards can in fact be worse than random actions in difficult environments requiring coordination. Rovers using D_i with full observability performed the best and rovers using $D_i(PO)$ performed well. In this more difficult domain, $D_i(PO)$ did not have significant learnability gains over D_i, and therefore, did not overcome the drop in factoredness. The $D_i(PO)$ results are still impressive though as they are obtained by using only about 3% of the information about the location of others rovers D_i has.

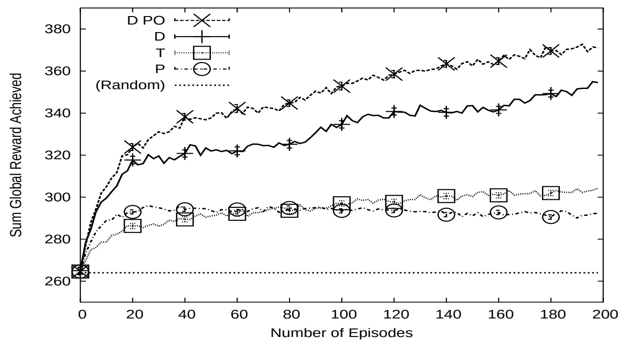

Figure 6. System performance in Static Environment. As predicted by the visualizations, agents using P_i have mediocre performance, agents T_i learn slowly and, $D_i(PO)$ retains enough factoredness to perform well.

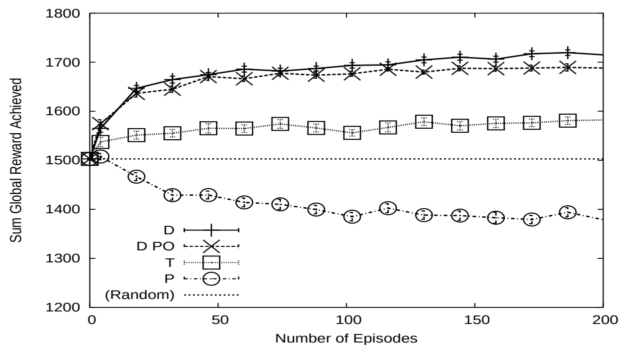

Figure 7. Results in Dynamic Environment. As predicted from the visualization, agents using T_i perform poorly, agents using P_i perform even worse as they learn the *wrong* actions, agents using D_i perform the best, and agents using $D_i(PO)$ perform quite well.

6. Conclusion

The effectiveness of agent rewards in promoting coordination in a complex multi-agent system is heavily domain dependent. In many cases, rewards or coordination mechanisms that work well in static environments perform poorly in dynamic environments. This paper shows that the visualization of two critical reward properties can dramatically accelerate and reduce the difficulties associated with choosing good agent rewards and coordination mechanism in difficult multi-agent problems. In addition the rewards can be modified to meet the computational and informational demands of a domain and then quickly validated. We demonstrate this capability by predicting the performance characteristics of a set of rewards in a noisy, continuous multi-rover domain, and show that some rewards that do work reasonably well in the static environment fall apart in the dynamic environment. This visualization method is one to two orders of magnitude faster than running a full learning simulation to validate the agent rewards. We used this visualization method to design and validate a reward based on a more computationally expensive reward. This reward only needed 3% of the observational capability of the full reward, but as predicted by the visualization performed nearly as well as the full reward in the dynamic environment.

References

[1] A. Agogino, C. Martin, and J. Ghosh. Visualization of radial basis function networks. In *Proceedings of International Joint Conference on Neural Networks*, Washington, DC, 1999.

[2] A. Agogino and K. Tumer. Efficient evaluation functions for multi-rover systems. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2004)*, pages 1–12, Seattle, WA, 2004.

[3] L. Baird and A. Moore. Gradient descent for general reinforcement learning. In *Advances in Neural Information Processing Systems (NIPS)*, pages 968–974, Cambridge, MA, 1999. The MIT Press.

[4] H. Bishof, A. Pinz, and W. G. Kropatsch. Visualization methods for neural networks. In *11th International Conference on Pattern Recognition*, pages 581–585, The Hague, Netherlands, 1992.

[5] C. B. Excelente-Toledo and N. R. Jennings. The dynamic selection of coordination mechanisms. *J. of Autonomous Agents and Multi-Agent Systems*, 9(1-2), 2004.

[6] P. Hoen and H. L. P. G. Redekar, V. Robu. Simulation and visualization of a market-based model for logistics management in transportation. In *Proceedings of the Third International Joint Conference on Autonomous Agents and Multi-Agent Systems*, pages 1218–1219, New York, NY, July 2004.

[7] M. J. Mataric. Coordination and learning in multi-robot systems. In *IEEE Intelligent Systems*, pages 6–8, March 1998.

[8] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, 1998.

[9] K. Tumer, A. Agogino, and D. Wolpert. Learning sequences of actions in collectives of autonomous agents. In *Proceedings of the First International Joint Conference on Autonomous Agents and Multi-Agent Systems*, pages 378–385, Bologna, Italy, July 2002.

[10] K. Tumer and D. Wolpert, editors. *Collectives and the Design of Complex Systems*. Springer, New York, 2004.

[11] J. Wejchert and G. Tesauro. Visualizing processes in neural networks. *IBM Journal of Research and Development*, 35:244–253, 1991.

[12] D. H. Wolpert and K. Tumer. Optimal payoff functions for members of collectives. *Advances in Complex Systems*, 4(2/3):265–279, 2001.