Analyzing and Visualizing Multi-Agent Rewards in Dynamic and Stochastic Domains

Adrian K. Agogino (adrian.k.agogino@nasa.gov)
University of California, Santa Cruz

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

Abstract. The ability to analyze the effectiveness of agent reward structures is critical to the successful design of multiagent learning algorithms. Though final system performance is the best indicator of the suitability of a given reward structure, it is often preferable to analyze the reward properties that lead to good system behavior (i.e., properties promoting coordination among the agents and providing agents with strong signal to noise ratios). This step is particularly helpful in continuous, dynamic, stochastic domains ill-suited to simple table backup schemes commonly used in TD(λ)/Q-learning where the effectiveness of the reward structure is difficult to distinguish from the effectiveness of the chosen learning algorithm.

In this paper, we present a new reward evaluation method that provides a visualization of the tradeoff between the level of 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 use this reward property visualization method to determine an effective reward without performing extensive simulations. We then test this method in both a static and a dynamic multi-rover learning domain where the agents have continuous state spaces and take noisy actions (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 good rewards, compared to running a full simulation. In addition, this method facilitates the design and analysis of new rewards tailored to the observational limitations of the domain, providing rewards that combine the best properties of traditional rewards.

1. Introduction

Creating high-performance reward structures that promote coordination is a critical step in designing successful multiagent learning algorithms [8, 15, 18, 21, 22, 26, 31]. Given an overall system reward that evaluates the performance of the full system, there are many ways in which local agent rewards can be derived. In very simple systems, agents may simply use the system reward directly. In more complex systems, we may wish to give agents more local rewards that increase their ability to learn good policies. Unfortunately this "reward engineering" task becomes more difficult in complex domains involving

continuous state spaces, and dynamic and stochastic environments. For example, reward structures that have been shown to perform well in static environments, do not necessarily lead to good system behavior in dynamic environments [10]. In order to determine the suitability of agent rewards to particular domains and environments, and to spur the design of agent rewards that lead to good system behavior, reward analysis methods that are independent of the domains and learning algorithms are needed.

In this paper, we present a reward analysis method that explicitly visualizes the desirable properties of a reward in both static and dynamic environments. This visualization method is designed to be used with a class of multiagent reward methods that has been shown to lead to good multiagent behavior in a large number of domains [4, 26, 28, 31]. The salient properties of this class of rewards are to assess:

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 its reward.

The first property promotes rewards such that when an agent takes an action that increases the value of its reward, the overall system reward also increases. The second property property promotes rewards that are heavily impacted by the agent's own actions. Note that in large systems, the full system reward does not have this property, as an agent's action has, on average, little impact of the full system.

This paper provides a visualization method that shows how a reward system performs with respect to these two reward properties. Using this visualization we can predict the reward performance in a given domain without the need for lengthy learning trials. In addition we can prune out problematic reward structures before they are implemented. Also even for reward structures in use, we can use this analysis to determine for what problems the reward may be unsuitable and avoid these situations. 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, analysis and visualization problem in a continuous multi-rover coordination domain where a set of rovers learn to navigate and collect information in an unknown environment based on their noisy sensor inputs [3]. Reinforcement learning and credit assignment are particularly challenging in this case because traditional table-based reinforcement learning methods such as Q-learning, TD(λ) and Sarsa learning are ill-suited to this domain [23]. 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 [26]. Therefore, having well tailored and computationally tractable agent rewards is particularly important in this domain.

Although in this paper we focus on the rover domain, both the reward design and the visualization approach have broad applicability. Indeed, the reward design framework used in this paper has been applied to many domains, including data routing over a telecommunication network [29], multiagent gridworld [26], congestion games (traffic toll lanes) [27, 28, 31] and optimization problems such as bin packing [32] and faulty device selection [24]. The strength and applicability of this visualization method stems from using projections from the agents' state space instead of using projections onto two-dimensional representations of the physical domain. For example, although we present a projection of the agent rewards' properties on the domain specific (x, y) plane in Figure 6 to demonstrate how and why some rewards are preferable to others, the more comprehensive visualizations for reward design are in Figures 5, 7 and 9. As such, this visualization method is not dependent on the domain having a natural projection to a 2-d plane, but on finding suitable axes from the agents' state space which do not need to correspond to any physical space in the domain. To that end, we present a projection method based on knowledge of the domain (Sections 4.1-4.2) as well as an automated projection (Section 4.3).

The major contribution of this paper is in providing a reward evaluation and visualization method for multiagent learning problems in stochastic 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. The visualization method is then used to determine which reward is best suited for 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 performance of these rewards in the rover domain.

2. Agent Coordination Rewards

The reward analysis and visualization method we present is based on properties derived for achieving good system level behavior in collectives, or cooperative multiagent systems with a well defined system level objective [31]. In this setting, each agent i takes actions to maximize its own agent reward g_i. The system performance is measured by the global reward G. For any agent i, 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 even 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

The visualization method we present is based on two agent reward properties that lead to good system behavior. First, we focus on aligning the agent rewards with the system reward, ensuring that agents that aim to achieve high values of their rewards do not end up harming the system. Second we focus on providing rewards that are sensitive to that particular agent's actions, ensuring that the agent receives sufficient "signal" to learn to optimize its own reward.

Formalizing the first property for an agent i, let us define the **degree of factoredness** (a generalization of factoredness presented in [4, 31, 27]) between the rewards g_i and G at state z, as[1] :

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

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 $x > 0$. The numerator counts the number of states where $g_i(z) - g_i(z')$ and $G(z) - G(z')$ have the same sign, and the denominator counts the total number of states. Intuitively, the degree of factoredness gives the fraction of states in which a change in the state of agent i has the same impact on g_i and G. A high degree of factoredness means that the agent reward g_i moves in the same direction (up or down) as the global reward G based on a change to the system state. A system in which all the agent rewards equal G has a degree of factoredness of 1. We call such a system "fully" factored.

[1] Note this factoredness concept is based on earlier work on multiagent coordination [26, 29, 31] and is not related to the concept of factored Markov Decision Processes [12, 13, 14, 20].

Formalizing the second property requires measuring the dependence of a reward on the state of a particular agent as opposed to the states of 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 state 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 state of agent i, i.e., the better the associated signal-to-noise ratio for agent i. Therefore, higher learnability means it is easier for agent i to take actions (changing its state) that maximize its reward. When learnability is too low, many other agents are affecting the reward, therefore it is hard for an agent to discern the affects of its actions from the actions of all of the other agents. 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 reward with high factoredness will have low learnability and a reward with high learnability will have low factoredness [31]. 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) each defined as:

$$T_i \equiv G(z) \tag{4}$$

$$P_i \equiv G(z_i) \tag{5}$$

$$D_i \equiv G(z) - G(z_{-i}). \tag{6}$$

T_i provides the full global reward to each agent. It is fully factored by definition and has been used successfully on multiagent problems

with few agents [9]. However, since each agent's reward depends on the states of all the other agents, it generally has poor learnability, a problem that gets progressively worse as the size of the system grows. This is because, when an agent's reward changes, it is difficult for the agent to determine whether that change was caused by its own action or by the actions of the other agents in the system.

In contrast to the global reward, the reward P_i is local and 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". However, depending on the domain, it may have a low degree of factoredness. Since it has high learnability, an agent can more easily achieve high values of this reward. However, since it does not necessarily have high factoredness, it is possible that an agent aiming to maximize this reward will take actions that will hurt the global reward.

The third reward, D_i, provides more of a balance between the degree of factoredness and learnability than do the other rewards. D_i has high factoredness, because $G(z_{-i})$, does not depend on agent i's states [31]. As a result, any impact an agent has on this reward comes from its impact on $G(z)$, which is the global reward. Hence, any actions that improve/deteriorate D_i also improve/deteriorate G. Furthermore, D_i usually has better learnability than does T_i, because subtracting out $G(z_{-i})$ removes some of the effects of other agents (i.e., noise) from agent i's reward. The reward D_i has been successfully used in multiple domains including packet routing over a data network [29], congestion games [31] and multiagent gridworlds [26]. While having good properties, this reward can be impractical to compute because it requires knowledge about z to compute $G(z_{-i})$. In such cases, the way in which the missing information will be handled (ignored, estimated) will determine the trade-off between factoredness and learnability.

3. Continuous Rover Problem

Because both the applicability and the effectiveness of the three rewards discussed above depend on the domain, reward analysis is critical before approaching any domain. In this section, we present the "Continuous Rover Problem," which will be used to illustrate the importance of visualization and proper reward selection in a difficult, noisy, continuous, multiagent domain. In this problem, multiple rovers try to observe points of interests (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:[2]

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

While any rover can observe any POI, as far as the global reward is concerned, only the closest observation counts[3]. The full system, or global reward is given by the aggregate value of all the POIs observed throughout the simulation:

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

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.

The rovers sense the world through eight continuous sensors and map those inputs into two outputs representing the x, y direction of motion. From a rover's point of view, the world is divided into four quadrants relative to the rover's current 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. The eight sensors provide the rover with a representation of their world based on the locations of POIs and other rovers. Note that this is an approximate representation of the world, as the location and number of rovers and POIs in each quadrant is reduced to an average number.

[2] 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.

[3] 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.

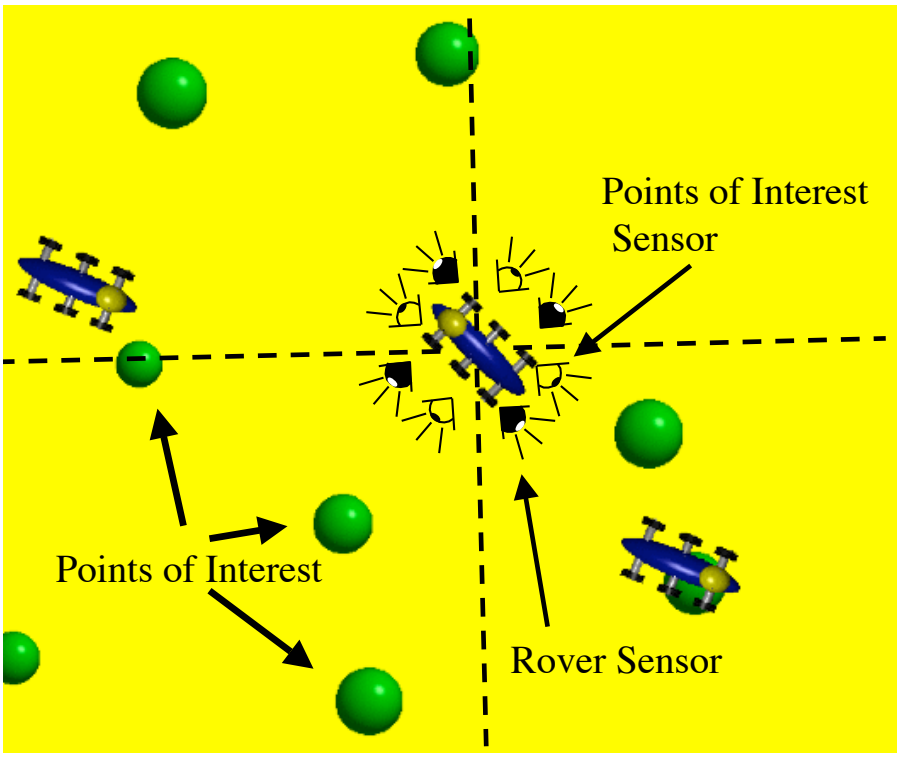

Figure 1. **Diagram of a Rover's Sensor Inputs.** The world is broken up into four quadrants relative to rover's position. In each quadrant one sensor senses points of interest, while the other sensor senses other rovers.

3.1. Simulation Set-up

With four quadrants and two sensors per quadrant, a rover has a total of eight continuous inputs. This eight dimensional sensor vector constitutes the state space for a rover. 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 (see Figure 2). 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 twice the maximum distance the rover can move in one time step: $dx = 2d(o_1 - 0.5)$ and $dy = 2d(o_2 - 0.5)$ 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 [5].

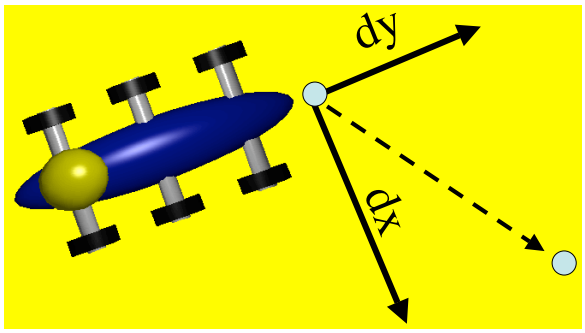

Figure 2. **Diagram of a Rover's Movement.** At each time step the rover has two continuous outputs (dx, dy) giving the magnitude of the motion in a two-dimensional plane relative to the rover's orientation.

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 global rewards, though each rover is trying to maximize its own agent reward, discussed below.

3.2. Rover Rewards

In this domain, the three agent rewards discussed in Section 2.2 are used. 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) which for this domain becomes:

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

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 in general fully factored and will often have a low degree of factoredness. Intuitively P_i and T_i offer opposite benefits, since T_i is by definition fully 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 fully factored like T_i . For the rover problem, directly applying Equation 6 to Equation 8 leads to:

$$D_i = \sum_j \frac{V_j}{\min_{i'} \delta(L_j, L_{i'})} - \sum_j \frac{V_j}{\min_{i' \neq i} \delta(L_j, L_{i'})}. \quad (12)$$

Note, only when rover i is the rover whose observation of a POI is used in the global reward will that difference be non-zero. Now, let $I_{j,i}$ be the indicator function that returns one if and only if rover i is the closest rover to POI j. Now, D_i becomes:

$$D_i = \sum_j I_{j,i} \left[\frac{V_j}{\delta(L_j, L_i)} - \frac{V_j}{\delta(L_j, L_{k_j})} \right] , \quad (13)$$

where k_j is the second closest rover to POI j. The second term of the D_i is equal to the value of all the information collected if rover i were 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 T_i, since T_i can be computed once and then broadcast to all the agents.

It is therefore imperative to analyze how a partially observable version of D_i will perform in this domain. The partially observable difference reward, $D_i(PO)$, differs from D_i in its estimation of whether it is the closest rover to a POI or not:

$$D_{i(PO)} = \sum_j \frac{V_j}{\min_{i' \in O_i^\rho} \delta(L_j, L_{i'})} - \sum_j \frac{V_j}{\min_{i' \in O_i^\rho, \neq i} \delta(L_j, L_{i'})} , \quad (14)$$

where O_i^ρ is the set of rovers that are within a radius of ρ from rover i, and thus are observable by rover i. When O_i^ρ only includes rover i, we set the second term of Equation 14 to zero, and $D_{i(PO)}$ reduces to P_i. In the visualization simulations below, we present the $D_{i(PO)}$ properties along with those of D_i, P_i and T_i.

3.3. Static and Dynamic Environments

In the static environment, the set of POIs remains fixed for all learning episodes. The POI distributions range from randomly distributed across the state to checkerboard patterns of uniform POIs. The results and insights gained from visualization are qualitatively similar in all cases. To illustrate the impact of visualization, we select the POI distribution depicted in Figure 3, which requires a moderate amount of coordination. The 15 POIs to the left have value 3.0, and the lone POI to the right has a value of 10.0.

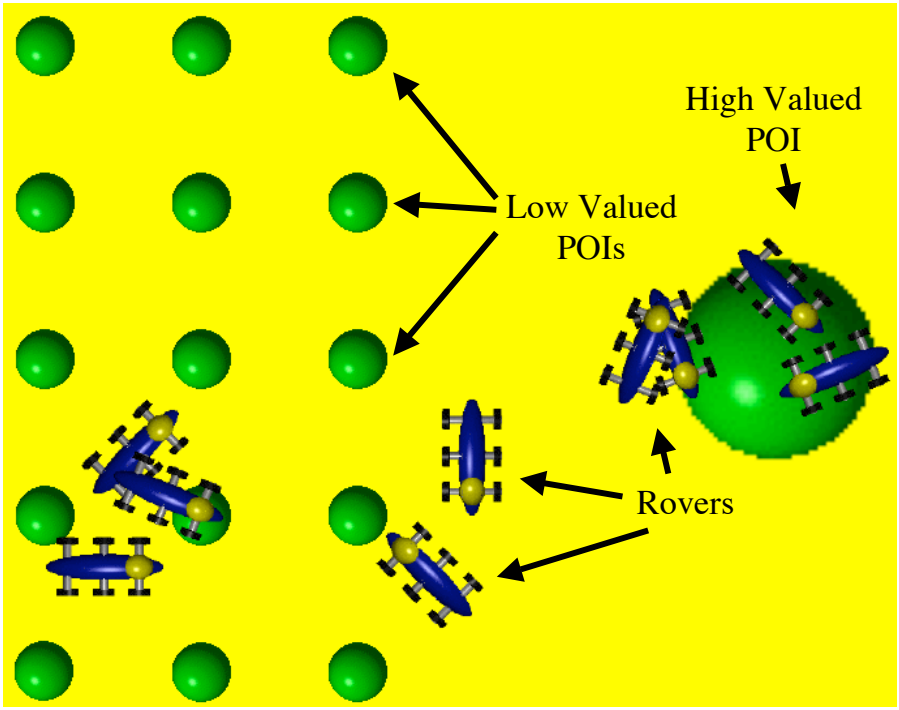

Figure 3. Diagram of Static Environment. Points of interests are at fixed locations for every episode.

In dynamic environments, the POI distribution changes every 15 time steps, and the rovers face a different configuration at each episode. In each episode, there are 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 can 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 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 [2, 6, 11, 16, 17, 30]. 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.

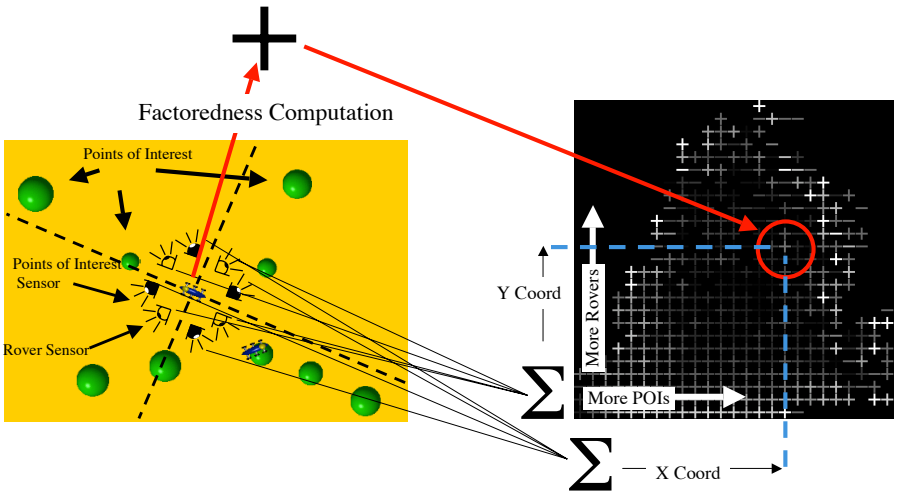

Figure 4. Factoredness Projection: The four POI sensors are combined to provide the x-axis and the four rover sensors are combined to provide the y-axis. The factoredness of a given reward is computed by sampling Equation 1 for each such point, and the results are plotted, varying from a bright "+" for fully factored to a dark spot for random, to a bright "-" for anti-factored.

The analysis starts by recording the states observed by agents taking a random set of actions [4]. For each reward, we compute the learnability and factoredness by sampling Equations 1-3. The learnability and

[4] States could also be recorded during learning, which could lead to different visualizations when the action distributions are significantly different than random and when an agent's factoredness strongly depends on the actions of other agents. However preliminary analysis in the rover domain has shown little difference.

factoredness values for each state are projected onto a two-dimensional plane. The projection is then broken up into fixed sized squares and all the values within a square are averaged. As with many visualization techniques, using the appropriate projection method is important to achieving good results. This paper presents results using a simple domain dependent projection and an automated projection using principal component analysis [19]. However, this method of visualization is adaptable with even more advanced projections methods, including non-linear projections [1, 7]. In addition the visualization can be an interactive process where the user switches between sets of projections to achieve the desired visualization. It is important to note that though the particular projection method is domain dependent, the impact of factoredness and learnability on the reward is applicable to a broad array of domains [25, 24, 29].

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 action influences 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-factored) are represented with a "-" symbol. The lighter the "-" symbol is, the more anti-factored is the corresponding reward. Figure 4 illustrates this process.

In this domain, the projection axes are formed using the eight sensor values used by the rovers. Note that many possible projection are possible. For most of the analysis in this paper, 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 visualization 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 5 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 under these 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, an expected result based on the fact that this reward only depends on the actions of agent i. This visualization implies that even though P_i is easy to learn, in many states it is likely to lead the agents to learn 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 entirely reversed for T_i in this environment. The T_i reward is by definition fully factored (except for states that have not been sampled, which show up as black in Figure 5), but has low learnability across the board. T_i has good learnability only on the right side of the visualization, corresponding to states where there are many POIs, meaning that the rover is generally 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 many POIs. D_i on the other hand has both high factoredness and high learnability. In this experiment ρ is set to the maximum distance a rover can move in one time step (10). 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 $D_{i(PO)}$ is not fully factored, the factoredness visualization shows that it still has good factoredness. In addition, the right side of the learnability visualization for D_i and $D_{i(PO)}$ (vertical rectangle marked 2 in Figure 5) shows that $D_{i(PO)}$ has higher learnability 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 dynamic environment where important regions keep shifting). This visualization shows how the rewards map to actions directly taken by the rovers. Figure 6 shows the factoredness for $D_{i(PO)}$ and P_i (on this projection, T_i

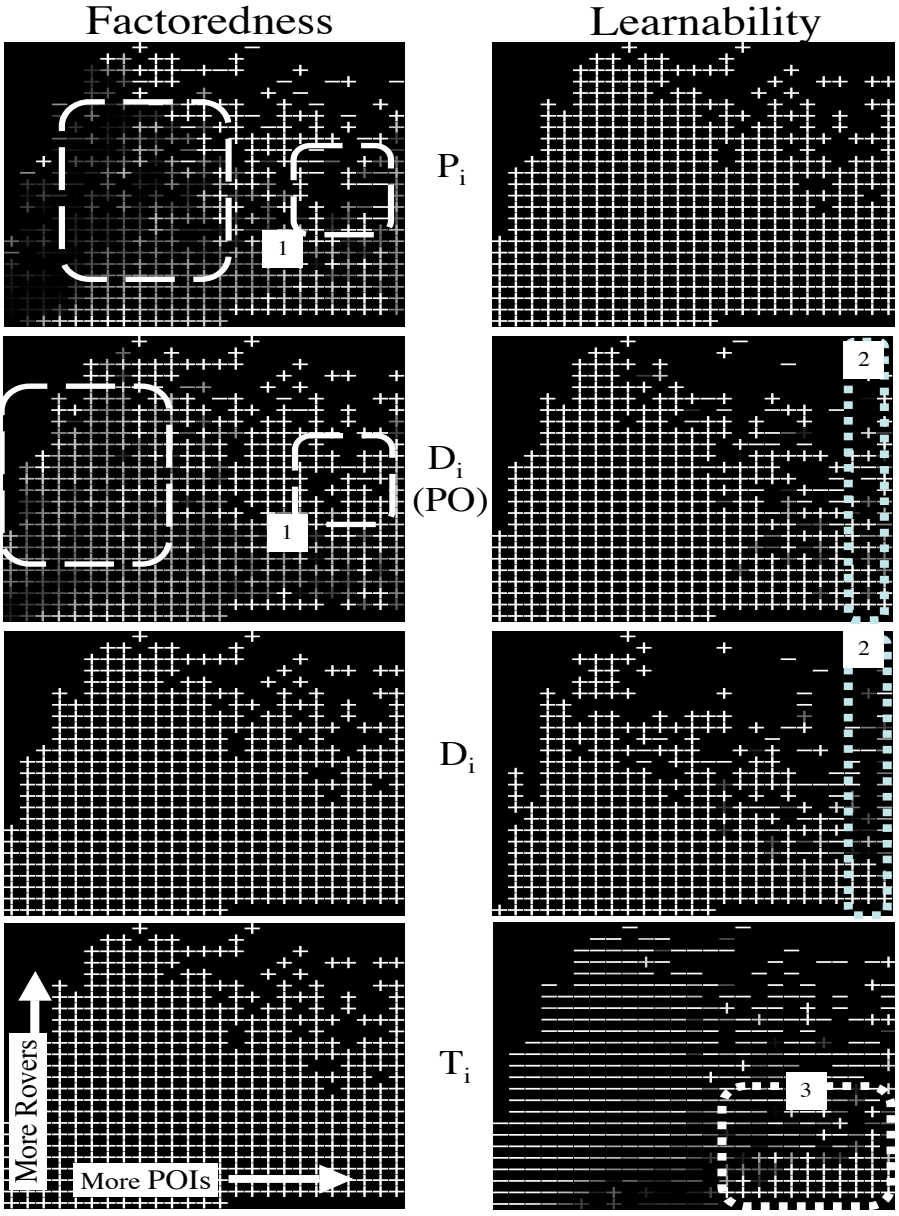

Figure 5. Factoredness and Learnability Visualization in Static Environment. First column shows factoredness of four rewards and second column shows their learnability. The visualization is a projection of an agent's state space, with increasing x values corresponding to states closer to POIs and increasing 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.

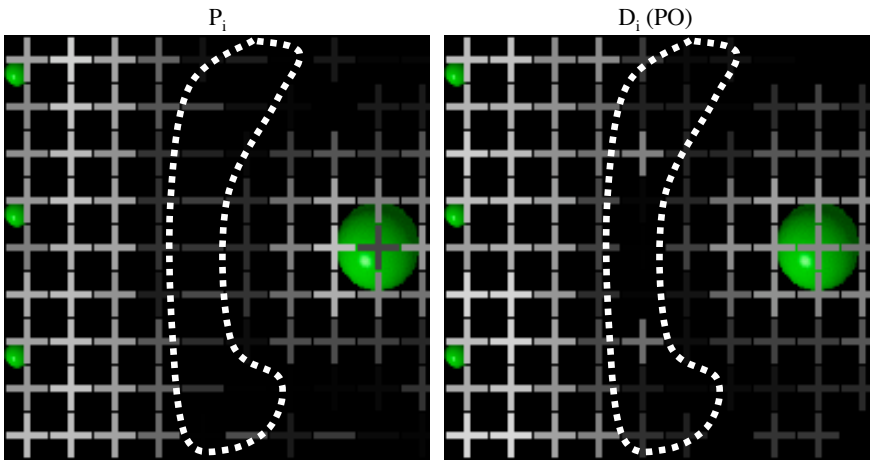

Figure 6. Factoredness Projected onto Domain Coordinates. Factoredness of P_i and $D_{i(PO)}$ is projected onto the x,y coordinates of the domain environment instead of onto the feature space used by rovers. "+" represents factoredness and "-" represents anti-factoredness. P_i has an anti-factored boundary preventing agents from moving from one region to the other.

and D_i are fully factored, meaning each square is a light "+"). Note that around the POIs, both rewards have high factoredness. 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 even though it does not have high factoredness in that region, $D_{i(PO)}$ has two "bridges" to cross this region and furthermore has low factoredness rather than being anti-factored in the rest of that region. This implies that $D_{i(PO)}$ will not perform poorly in this domain.

4.2. Visualization in Dynamic Environments

Figure 7 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 only marginally better than a random algorithm. P_i only has high factoredness 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

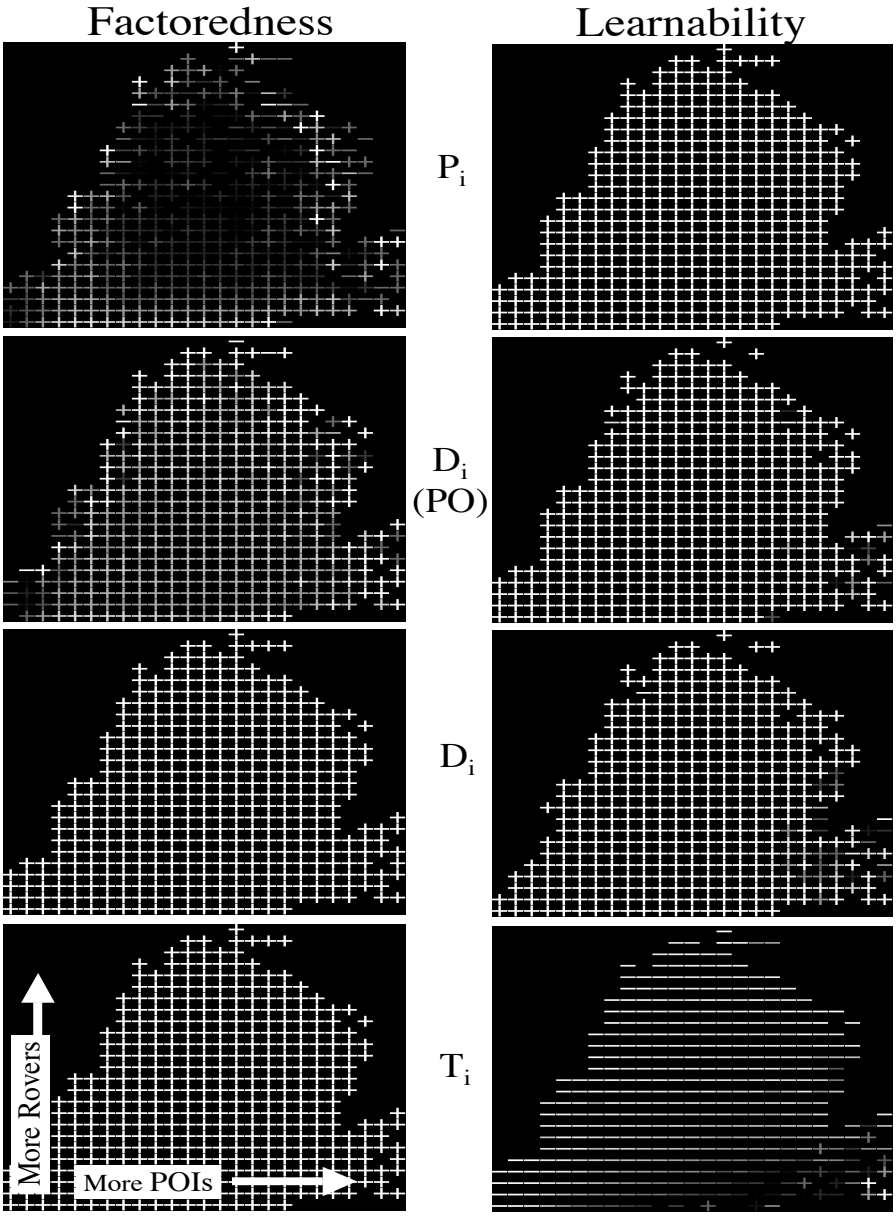

Figure 7. Factoredness and Learnability Visualization in Dynamic Environments. First column shows factoredness of the four rewards and second column shows their learnability. The visualization is a projection of an agent's state space. The visualizations show that P_i has very low factoredness and T_i has very low learnability. $D_{i(PO)}$ (computed with partial observability) still has high factoredness.

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 has both high learnability and high factoredness 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 compute $D_{i(PO)}$ 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.

4.3. Automated Projections in Dynamic Environments

The previous visualizations were made using hand-crafted projections that fit well with the rover domain. More generally such use of hand-crafted projections is appropriate in domains where a small set of informative features used for the state-space of the agents are known, and these features separate the domain into distinct regions. However, it is often desirable to have the ability to automatically create two-dimensional projections. This is especially true in domains with high dimensional state-spaces, where good two-dimensional projections may be non-obvious. In this section we use principal components to automatically create projections that can be used with the visualization.

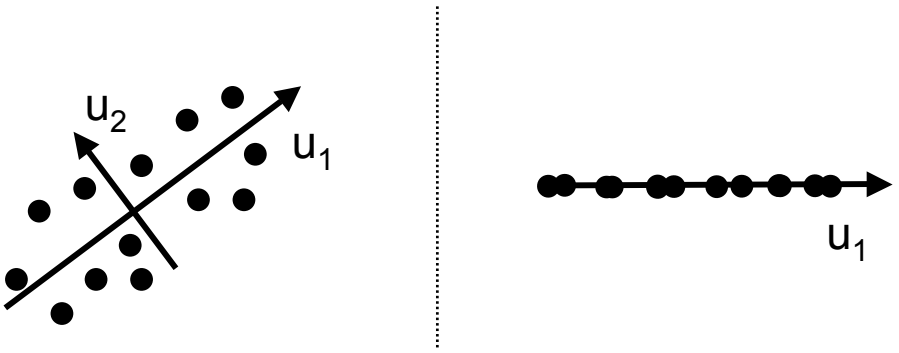

Figure 8. Principal Component Projections. Here data is projected from two dimensions on to one dimension using the first principal component. This component corresponds to the direction where the data is spread out the most.

Principal component projections are linear projections based on the principal components of the data [19]. To find the principal components projection axis for each rover, we first compute an $8x8$ covariance

matrix, Σ, based on the sensor inputs. Each element i, j of Σ is the covariance of sensor i with sensor j and is computed from all of the sensor inputs recorded during learning. The principal projections are then found by computing the eigenvectors of Σ. The principal components corresponding to the largest eigenvalues are along the directions where the data is spread out the most (see Figure 8). For the rover problem we use the first two principal components for the visualization.

The visualization resulting from principal component projections is shown in Figure 9. From this visualization, we come to similar conclusions as the previous visualization in Figure 7. Most importantly this visualization confirms the conclusion of the previous visualizations that the P_i reward has low factoredness and that the T_i reward has low learnability.

Though they provide a direct way to select the projection axis, the automated projects are more difficult to interpret directly. For instance the factoredness visualization for P_i shows a ring of factored regions around the edge of the visualization, with low factoredness towards the center of the visualization. From this diagram alone, it is difficult to tell how important these regions are. To help with this analysis we can plot the projected density of the rovers using the same two principal components as shown in Figure 10. This figure shows us where in the projected space the rovers actually spent most of their time. An inspection of the figure suggests that the rovers in fact spent most of their time in locations where P_i had the least factoredness, showing that this reward is especially bad in this domain.

Even in situations when regions in the principal component projection cannot be interpreted directly, the projection still has value when small subsets of the visualization are particularly interesting. Actual rover data corresponding to these interesting regions can then be extracted from all the less interesting rover data and can be analyzed more carefully. For instance if we found that for a particular reward a region of the visualization had very low factoredness, we could look at the locations of all the rovers that were projected to that region. We might then want to avoid using the reward for rovers likely to be in those locations.

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. In the experiments we measure the performance of the rovers over two hundred learning trials.

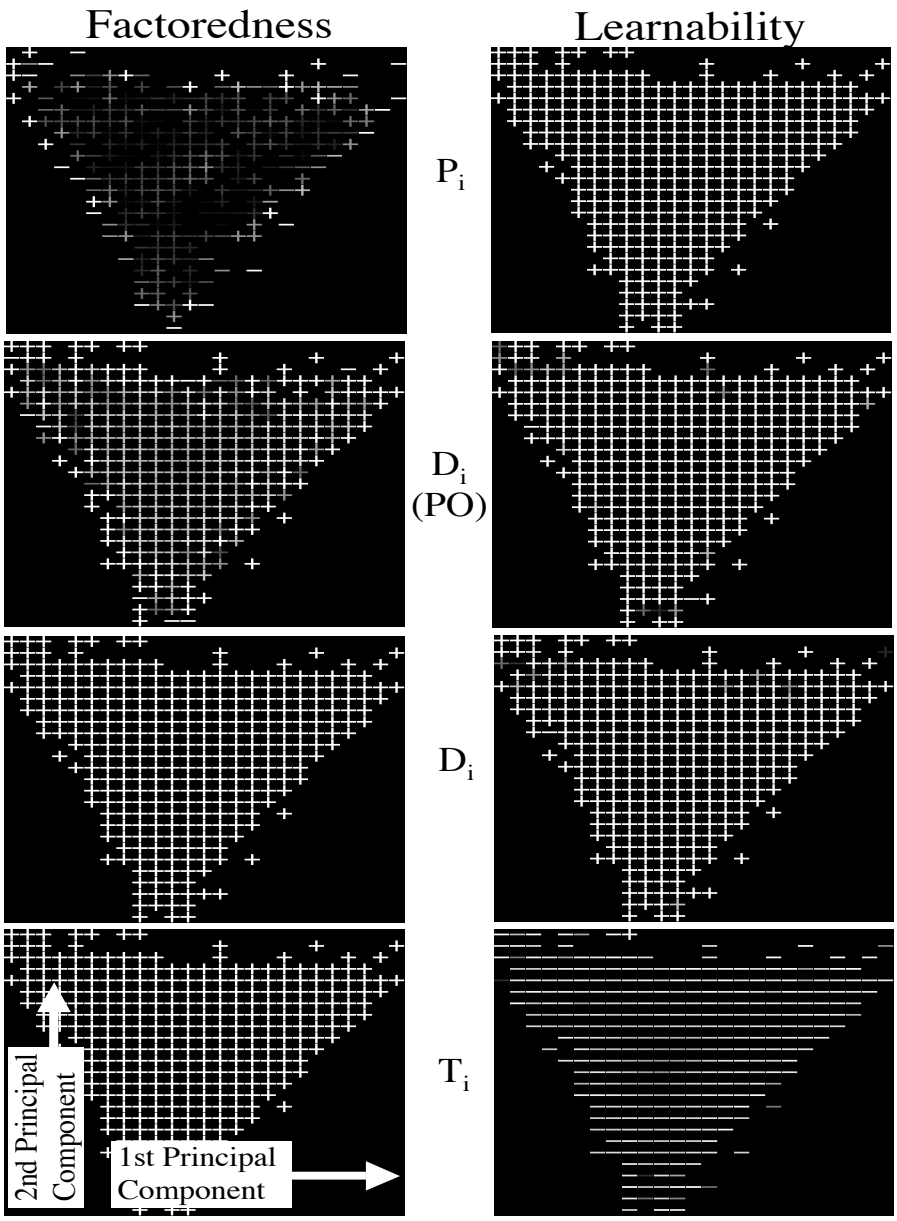

Figure 9. Factoredness and Learnability Visualization in Dynamic Environments Using Principal Components. Eight dimensional state space for rovers is projected on to two dimensions using principal components. First column shows factoredness of the four rewards and second column shows their learnability. 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.

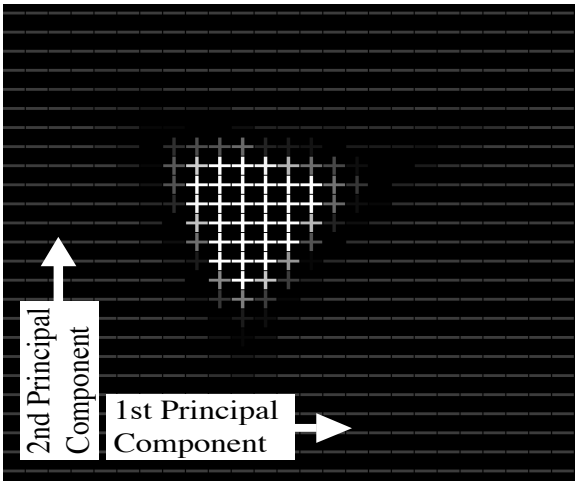

Figure 10. Density of Rovers State Spaces Projected Through Principal Components. Pluses are regions of high density. Minuses regions of low density. Rovers spend most of their time in the middle of the projected space. This region corresponds to where the P_i reward has lowest factoredness.

We then compile results averaging over thirty trials [5]. The experiments confirm the expectation obtained from the factoredness and learnability visualizations.

Figure 11 shows results from the static environment. The rovers using P_i learn quickly, but do not converge to good solutions. This result is consistent with the high learnability/low factoredness properties of P_i that are apparent in the visualizations. In contrast agents using T_i are able to keep improving their performance through learning, and are 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 if quick learning is needed. As expected, rovers using D_i with full observability perform very well, since D_i has both high learnability and is fully factored. More interestingly, the rovers using $D_{i(PO)}$ perform even better even 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 a remarkable (and counter-intuitive) result, since $D_i(PO)$ is in fact significantly easier to compute than D_i, and has access to less information about the state of the system.

[5] This represents a total of six thousand trials performed to get statistically reliable results, while the visualizations were computed with fifty trials. This shows that the visualizations can assess potential reward performance over two orders of magnitudes faster than actually running the full simulation.

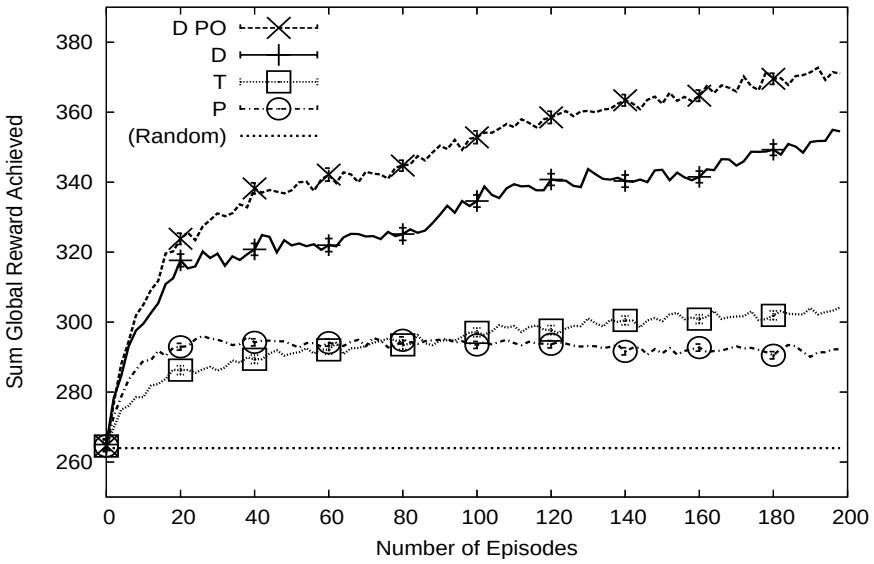

Figure 11. System performance in Static Environment. As predicted by the visualizations, agents using P_i have mediocre performance, agents T_i learn slowly and, $D_{i(PO)}$ retains enough factoredness to perform well.

Figure 12 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 result confirms the intuition that rewards with high learnability and low factoredness can in fact be worse than random actions in difficult environments requiring coordination. Rovers using D_i with full observability perform the best and rovers using $D_{i(PO)}$ perform well. In this more difficult domain, $D_{i(PO)}$ does not have significant learnability gains over D_i, and therefore, does 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 that D_i has about the location of others rovers.

6. Discussion

The effectiveness of agent reward structures in promoting good system level behavior in a complex multiagent system is heavily domain dependent. Predicting the behavior of such a multiagent system is difficult at best, and in general improving the reward structure relies on experimenting with different rewards and performing extensive simulations to

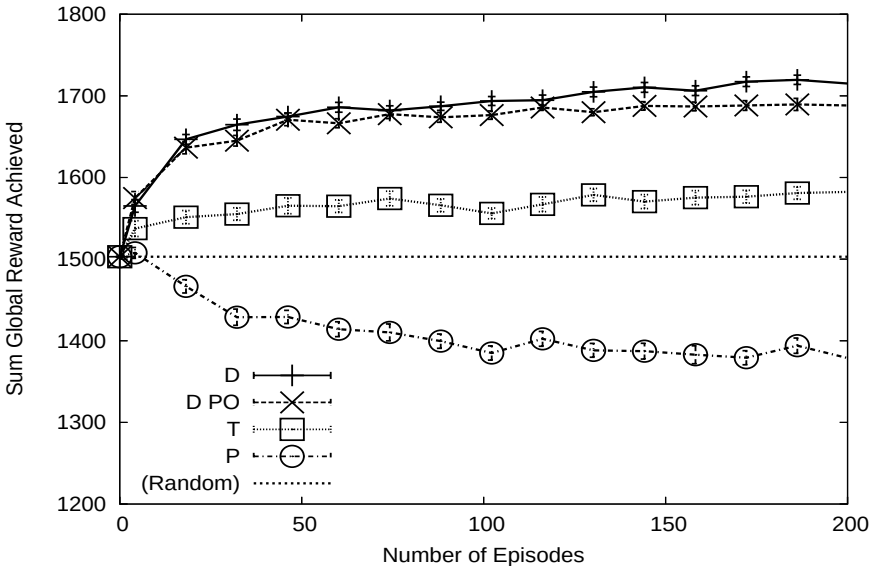

Figure 12. Results in Dynamic Environment. As predicted from the visualization, agents using T_i perform poorly, agents using P_i perform even worse as they learn the *wrong* actions, agents using D_i perform the best, and agents using $D_{i(PO)}$ perform quite well.

assess the merits of each different reward structure. Furthermore, using shortcuts such as using a static domain to test a reward structure to be used in a dynamic domain may lead to poor performance, as rewards or coordination mechanisms that work well in static environments may perform poorly in dynamic environments. To address these issues, in this paper we introduce a reward analysis method showing that the visualization of two critical reward properties can dramatically accelerate reward design and reduce the difficulties associated with choosing good agent rewards in difficult multiagent problems.

This reward visualization is based on projecting two reward properties (*factoredness*, measuring reward alignment; and *learnability*, measuring reward sensitivity to an agent's actions) into a domain dependent chart. We show both a handcrafted projection and an automated projection based on the principal components of an agent's state space. Both sets of results predicted the same performance ordering among the tested rewards, and predicted that some rewards that work well in a static domain fall apart in a dynamic domain. The simulation results confirmed those predictions. The particular projection methods used are by necessity domain dependent. However, the benefits of designing rewards that have high factoredness and learnability are broadly applicable.

In addition, this analysis and visualization method allows the rewards to be modified to meet the computational and observational demands of a domain. We demonstrate this capability by predicting the performance characteristics of a reward based only on 3% of the available information and showed that this reward performed remarkably well. Indeed, the partially observable D_i performed better than the fully observable D_i in the static environments, and as predicted by the visualization, performed nearly as well as the fully observable reward in the dynamic environment. Furthermore, the visualization plots were obtained up to two orders of magnitude faster (in the case of the dynamic environment) than running a full learning simulation to validate the agent rewards.

Finally, in this paper the visualization plots were computed based on sampling the states uniformly. Though this method has the advantage of being independent of the learning algorithm, it can distort the factoredness and learnability plots in two ways. First, the likelihood of a state being visited is not taken into account in generating the plots. As shown in Figure 10, determining which states are likely to be visited will change the interpretation of these plots. Second, the dynamics of the system are considered implicitly, not explicitly in these visualization methods. By sampling the state space, we gather information about each state, but do not consider how an agent proceeds from one state to the next. Including such information requires knowledge about the learning algorithm, and leads to a three way factoredness (system reward, agent reward, agent learning algorithm) which can be used to select/tune learning algorithms for a particular agent reward selection. We are currently exploring these issues and particularly focusing on visualizing the time-dependent dynamics created by the factoredness issues. We are focusing on the use of "time slices" as representations of factoredness and assessing how an agent's reward may be "factored through time". We anticipate this analysis to provide agent reward and agent learning algorithms better tuned to the particular domain and yield further improvements over those we have so far achieved.

References

1. Agogino, A., C. Martin, and J. Ghosh: 1998, 'Principal Curve Classifier - A Nonlinear Approach to Pattern Classification'. In: *Proceedings of International Joint Conference on Neural Networks*. Anchorage, Alaska.
2. Agogino, A., C. Martin, and J. Ghosh: 1999, 'Visualization of Radial Basis Function Networks'. In: *Proceedings of International Joint Conference on Neural Networks*. Washington, DC.

3. Agogino, A. and K. Tumer: 2004, 'Efficient Evaluation Functions for Multi-Rover Systems'. In: *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2004)*. Seattle, WA, pp. 1–12.
4. Agogino, A. and K. Tumer: 2005, 'Multi Agent Reward Analysis for Learning in Noisy Domains'. In: *Proceedings of the Fourth International Joint Conference on Autonomous Agents and Multi-Agent Systems*. Utrecht, Netherlands.
5. Baird, L. and A. Moore: 1999, 'Gradient Descent for General Reinforcement Learning'. In: *Advances in Neural Information Processing Systems (NIPS)*. Cambridge, MA, pp. 968–974.
6. Bishof, H., A. Pinz, and W. G. Kropatsch: 1992, 'Visualization Methods for Neural Networks'. In: *11th International Conference on Pattern Recognition*. The Hague, Netherlands, pp. 581–585.
7. Bishop, C. M.: 1995, *Neural Networks for Pattern Recognition*. New York: Oxford University Press.
8. Chalkiadakis, G. and C. Boutilier: 2003, 'Coordination in Multiagent Reinforcement Learning: A Bayesian Approach'. In: *Proceedings of the Second International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS-03)*. Melbourne, Australia.
9. Crites, R. H. and A. G. Barto: 1996, 'Improving Elevator Performance using Reinforcement Learning'. In: D. S. Touretzky, M. C. Mozer, and M. E. Hasselmo (eds.): *Advances in Neural Information Processing Systems - 8*. pp. 1017–1023.
10. Excelente-Toledo, C. B. and N. R. Jennings: 2004, 'The Dynamic Selection of Coordination Mechanisms'. *J. of Autonomous Agents and Multi-Agent Systems* **9**(1-2).
11. Gallagher, M. and T. Downs: 1997, 'Visualization of Learning in Neural Networks Using Principal Component Analysis'. In: *International Conference on Computational Intelligence and Multimedia Applications*. pp. 327–331.
12. Guestrin, C., M. Hauskrecht, and B. Kveton: 2004, 'Solving Factored MDPs with continuous and discrete variables'. In: *Proceedings of the 20th Conference on Uncertainty in Artificial Intelligence*. pp. 235–242.
13. Guestrin, C., D. Koller, and R. Parr: 2001a, 'Max-norm projections for factored MDPs'. In: *Proceedings of the International Joint Conference on Artificial Intelligence*.
14. Guestrin, C., D. Koller, and R. Parr: 2001b, 'Multiagent planning with factored MDPs'. In: *NIPS - 14*.
15. Guestrin, C., M. Lagoudakis, and R. Parr: 2002, 'Coordinated Reinforcement Learning'. In: *Proceedings of the 19th International Conference on Machine Learning*.
16. Hinton, G.: 1986, 'Connectionist Learning Procedures'. *Artificial Intelligence* **40**, 185–234.
17. Hoen, P. and H. L. P. G. Redekar, V. Robu: 2004, '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*. New York, NY, pp. 1218–1219.
18. Hu, J. and M. P. Wellman: 1998, 'Multiagent Reinforcement Learning: Theoretical Framework and an Algorithm'. In: *Proceedings of the Fifteenth International Conference on Machine Learning*. pp. 242–250.
19. Jolliffe, I.: 2002, *Principal Component Analysis*. New York: Springer, second edition.

20. Kearns, M. and D. Koller: 1999, 'Efficient Reinforcement learning in Factored MDPs'. In: *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*. pp. 740–747.
21. Mataric, M. J.: 1998, 'Coordination and Learning in Multi-Robot Systems'. In: *IEEE Intelligent Systems*. pp. 6–8.
22. Stone, P. and M. Veloso: 2000, 'Multiagent Systems: A Survey from a Machine Learning Perspective'. *Autonomous Robots* **8**(3).
23. Sutton, R. S. and A. G. Barto: 1998, *Reinforcement Learning: An Introduction*. Cambridge, MA: MIT Press.
24. Tumer, K.: 2005, 'Designing Agent utilities for Coordinated, Scalable and Robust Multi-Agent Systems'. In: P. Scerri, R. Mailler, and R. Vincent (eds.): *Challenges in the Coordination of Large Scale Multiagent Sy stems*. Springer. to appear.
25. Tumer, K. and A. Agogino: 2007, 'Distributed Agent-Based Air Traffic Flow Management'. In: *Proceedings of the Sixth International Joint Conference on Autonomous Agents and Multi-Agent Systems*. Honolulu,HI, pp. 330–337. **Best paper award**.
26. Tumer, K., A. Agogino, and D. Wolpert: 2002, 'Learning Sequences of Actions in Collectives of Autonomous Agents'. In: *Proceedings of the First International Joint Conference on Autonomous Agents and Multi-Agent Systems*. Bologna, Italy, pp. 378–385.
27. Tumer, K. and D. Wolpert (eds.): 2004a, *Collectives and the Design of Complex Systems*. New York: Springer.
28. Tumer, K. and D. Wolpert: 2004b, 'A Survey of Collectives'. In: *Collectives and the Design of Complex Systems*. Springer, pp. 1,42.
29. Tumer, K. and D. H. Wolpert: 2000, 'Collective Intelligence and Braess' Paradox'. In: *Proceedings of the Seventeeth National Conference on Artificial Intelligence*. pp. 104–109.
30. Wejchert, J. and G. Tesauro: 1991, 'Visualizing Processes in Neural Networks'. *IBM Journal of Research and Development* **35**, 244–253.
31. Wolpert, D. H. and K. Tumer: 2001, 'Optimal Payoff Functions for Members of Collectives'. *Advances in Complex Systems* **4**(2/3), 265–279.
32. Wolpert, D. H., K. Tumer, and E. Bandari: 2004, 'Improving Search Algorithms by Using Intelligent Coordinates'. *Physical Review E* **69**, 017701.