Gaussian Processes as Multiagent Reward Models

Gaurav Dixit
dixitg@oregonstate.edu
Oregon State University
Corvallis, Oregon

Stéphane Airiau
stephane.airiau@dauphine.fr
LAMSADE, CNRS, Université
Paris-Dauphine, Université PSL
Paris

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

ABSTRACT

In multiagent problems that require complex joint actions, reward shaping methods yield good behavior by incentivizing the agents' potentially valuable actions. However, reward shaping often requires access to the functional form of the reward function and the global state of the system. In this work, we introduce the Exploratory Gaussian Reward (EGR), a new reward model that creates optimistic stepping stone rewards linking the agents potentially good actions to the desired joint action. EGR models the system reward as a Gaussian Process to leverage the inherent uncertainty in reward estimates that push agents to explore unobserved state space. In the tightly coupled rover coordination problem, we show that EGR significantly outperforms a neural network approximation baseline and is comparable to the system with access to the functional form of the global reward. Finally, we demonstrate how EGR improves performance over other reward shaping methods by forcing agents to explore and escape local optima.

KEYWORDS

Multi-agent learning; Reward structures for learning; Learning agent-to-agent interactions; Reinforcement learning

ACM Reference Format:
Gaurav Dixit, Stéphane Airiau, and Kagan Tumer. 2020. Gaussian Processes as Multiagent Reward Models. In *Proc. of the 19th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2020), Auckland, New Zealand, May 9–13, 2020,* IFAAMAS, 9 pages.

1 INTRODUCTION

Multiagent coordination has been successfully applied to a wide variety of coordination tasks such as search and rescue [31], air traffic control [19, 28], exploration [4, 11, 17], bandwidth management [3, 15], and satellite configuration [12]. The credit assignment problem of how to assess the contribution of an individual agent based on system performance is a key problem in such settings. Reward shaping partially addresses this problem by tuning an agent's reward to a (potentially local) signal that is sensitive to that agent's actions [32].

However, many reward shaping methods either require some domain knowledge [14, 23] or need the agents to stumble upon the right joint action by "accident" to then observe the reward that incentivizes those actions in the future. Recent work to provide "stepping stone" rewards to encourage agents to discover desirable joint-actions focused on agents considering the impact of hypothetical partners when needed [25]. Such shaped rewards guide agents to potentially valuable actions even if other agents do not yet take the complementary actions. However, to generate stepping stone rewards, agents need access to the functional form of the system reward function to evaluate the effect of hypothetical partners. That requirement breaks down in many real world domains where the functional form of the system reward is either not known or not available to the agents.

Using a function approximator to learn the state-action to system performance mapping offers a potential solution [10]. But for such an approach to work, the agents need to generate statistically useful training data by sampling the state space to enable the function approximator (e.g. a neural network) to learn the mapping. Unfortunately, in tightly-coupled domains, where the joint action of multiple agents is required for achieving system objectives, such data is not available because agents never stumble upon the correct joint actions in the first place. For example, consider the problem of ascertaining there are no intruders in a 2D world with sparsely placed polygons. The task is only successful if all the faces of a polygon are observed simultaneously by the agents. Two agents should be in proximity and actively observing for a line, three for a triangle, four for a square and so on. In such problems, agents learning to coordinate requires all agents to stumble upon simultaneous face observations, which is highly unlikely in all but the simplest cases.

In this work, we introduce the Exploratory Gaussian Reward (EGR), which models the system reward as a Gaussian Process. EGR enables the use of shaped rewards that need estimating system performance in the presence of hypothetical agents, and leverages the inherent uncertainty in reward estimates associated with unobserved state space to build a robust state-to-reward mapping. In addition, EGR incorporates the confidence of the prediction in its estimate, leading difficult-to-stumble-into regions of the state space to have a low confidence and thus a high potential for rewards. This creates optimistic stepping stone rewards whose uncertainty over future rewards acts as a natural driver for exploration that pushes agents towards taking potentially beneficial joint-actions.

The contributions of this work are to:

- Introduce a mechanism to create reward models that induce optimistic stepping stone rewards without access to the functional form of the system reward.
- Improve the system performance with a probabilistic method that leverages the uncertainly in reward estimates to drive exploration in multiagent systems with high coordination requirements.

We show that EGR successfully guides agents to learn coordination strategies without access to the functional form of the system reward in the tightly coupled multi-rover exploration problem. As

Proc. of the 19th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2020), B. An, N. Yorke-Smith, A. El Fallah Seghrouchni, G. Sukthankar (eds.), May 9–13, 2020, Auckland, New Zealand.

a baseline, we compare the performance of EGR to local approximations of the system reward using a neural network [10]. EGR significantly outperforms local approximations for a tight rover coordination constraint of 3. We also show that EGR achieves superior performance for up to a coordination requirement of 9 rovers compared to the performance of the baseline. Finally, we show that EGR performs better than methods with access to the functional form of the system reward in tasks where rovers must sufficiently explore the state space to avoid getting stuck in local optima.

2 BACKGROUND

2.1 Multiagent Reinforcement Learning

Multiagent reinforcement learning is a natural extension of reinforcement learning where multiple cooperative, competitive or mixed goal oriented agents must work together to achieve a system objective [7, 8]. It is challenging due to the inherent non-stationarity of the environment created by multiple agents learning simultaneously. Most methods tackle the problem by either treating multiagent systems as a group of independent learners or by using a centralized architecture. Independent Q-learning agents for example, will often excel at tasks that can be broken down to discreet sub tasks but will fail if a task requires coordination and agent coupling [27]. For example, a cloud based service may deploy multiple servers to distribute the load of incoming requests. Delegating client requests to servers randomly will relieve the load and will help to scale. Although this is an improvement over a single server based system, the throughput of the system could be greatly improved by incorporating a load balancing mechanism. Often, this will be implemented as a centralized solution. Centralized architectures in multiagent reinforcement learning rely on using joint states and actions of all the agents in the system to learn coordination policies [18, 22]. These methods don't scale when agents are added or removed from the system because their architecture is a function of the properties of the current agent pool.

2.2 Reward Shaping

In multiagent systems that require coordination among agents, agent-to-agent interactions are very crucial to the overall system performance [20, 26, 30]. Agents need sufficient feedback to learn their contribution towards the system performance. Reward shaping is a method to create surrogate rewards to enable credit assignment in multiagent systems [5, 32]. In systems with coordination requirements, these local rewards act as "stepping stones" to guide agents to potentially valuable system rewards. Although a crucial tool, shaped rewards must be designed and engineered to make sure they align with the true system rewards. Often times, if they are not carefully shaped, agents will learn to exploit them and disregard the actual system performance.

2.3 Difference Evaluations

A key difficulty in multiagent learning is credit assignment to individual agents in the system when other agents are learning simultaneously. However, learning local policies has been show to improve performance in systems with low coordination requirements [6, 24, 26]. This is achieved by shaping local agent rewards such that they are aligned with the system performance; A higher local reward signal corresponds to an improvement in the overall system performance. Difference evaluations (Eqn. 1) are reward shaping methods that align the local agent rewards by providing agents with shaped rewards that are commensurate to their impact on the system performance [1]. In particular, difference evaluation computes an agent's impact on the system by leveraging a counterfactual action that effectively removes the agent from the system and evaluates the system performance. The change in reward when the agent was removed provides the agent with a clear feedback of its action. The counterfactual action could either be a "null" action meaning the agent does nothing or it could be a default action sampled from the agent's action space.

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

In Eqn. 1, z represents the current joint-state of the agents in the system, $D_i(z)$ is the shaped difference reward received by agent i, $G(z)$ is the system reward, and $G(z_{-i} \cup c_i)$ is the system reward with a counterfactual action, c_i, taken in place of agent i's action.

Difference evaluations have shown to vastly improve the system performance in loosely-coupled systems [1, 2, 29]. However in tightly-coupled systems, difference evaluations struggle to provide enough feedback to agents to promote teaming and coordination. For example, in a task that requires 3 agents to take complementary actions simultaneously, a single agent taking the right action does not generate a reward signal. Furthermore, removing this agent to gauge the impact of the agent's beneficiary action provides no feedback since the coupling requirement of the task is still unsatisfied. In such domains, agents would need to stumble upon good coordinating states to get a reward signal from the system evaluation. Even in loosely coupled domains, if the rewards are sparse, difference evaluation are not capable of shaping the rewards sufficiently.

2.4 D_{++} Evaluation

Difference evaluations struggle in tasks that require high coupling due to reward sparsity. To guide agents to potentially promising coordination strategies, D_{++} structural credit assignment introduces "stepping stone" rewards. These are local agent rewards that reinforce actions that might lead to coordination if other agents took complementary actions. In practise, an agent that performs a promising action receives a fraction of the reward it would have gotten if other agents would have cooperated. Unlike Difference evaluations where an agent is removed from the system, D_{++} introduces counterfactual agents which are copies of the current agent and occupy the same position as the agent.

$$D_{++}(i, n, z) = \frac{G(z_{+(\cup_{j=1,\ldots,n}}) - G(z)}{n} \tag{2}$$

In Enq. 2, n represents the number of counterfactual partners added, $D_{++}(i, n)$ represents the reward received by agent i with n counterfactual partners, and $G(z_{+(\cup_{j=1,\ldots,n}})$ represents what the system reward would be if there were n other agents in the system coordinating with the current agent i. A key drawback that restricts the application of D_{++} evaluation to most multiagent systems is its dependence on the functional form of the system reward. The system reward function is often inaccessible to individual agents or intractable. Even in systems where the functional form of the reward is available, computing D_{++} rewards for all agents can be

prohibitively expensive as it is computed for all agents locally and needs several evaluations of the reward function at every learning step.

2.5 Potential Based Reward Shaping

Potential Based Reward Shaping (PBRS)[14, 23] addresses the problem of sparse rewards in multiagent coordination by injecting stepping stones rewards at key states that guide agents towards desirable behavior. Unlike Difference evaluation, PBRS can be applied to multiagent systems when the functional form of the system reward is inaccessible and can create local stepping stone rewards for agents. PBRS modifies the original reward by adding in the difference of potential ϕ between the current state and the next state of the system.

$$PBRS\_reward = reward + \gamma\phi(s') - \phi(s)$$

A major drawback of Potential-Based reward shaping is the requirement of domain knowledge to design the potential function ϕ over system states. Incorrectly and overly engineered rewards often lead to undesired behavior when the overall system performance becomes misaligned with the stepping stone rewards [13].

2.6 Reward Approximation

To address the problem of creating stepping stone rewards when the system reward function is inaccessible, recent methods focus on ways to approximate it or to create surrogate rewards that reinforce exploration. Recently it was shown that a fully connected neural network could be used to locally approximate the system reward function and compute difference evaluation to achieve agent coordination in multiagent rover exploration and the El Farol Bar problem [10]. Agents in the system use a broadcast value of the system reward $G(z)$ to create a local approximation. At every learning step, agents update their approximated $\hat{G}(z)$ using the update rule in Eqn. 3.

$$\hat{G}(z) \leftarrow (1-\alpha)\hat{G}(z) + \alpha G(z) \tag{3}$$

where, α is the learning rate of the approximator. The approximate difference evaluation is then computed using Eqn. 4.

$$\hat{D}(z) = \hat{G}(z) - \hat{G}_i(c_i) \tag{4}$$

for agent i taking a default counterfactual action c_i. Agents do not need to monitor actions of other agents in the system since they use their locally approximated $\hat{G}_i(c_i)$ for difference evaluations. For loose coordination requirements where agents will stumble into coordinating states, a function approximator can model the system reward signal after sufficient observations.

However, to satisfy tighter coordination constraints which require more than 2 agents to simultaneously take complimentary actions, stumbling upon these state space regions becomes highly improbable. Without sufficient observations, the system reward cannot be approximated to compute D_{++} rewards.

3 ROVER COORDINATION PROBLEM

In this work, we use a tightly-coupled multiagent variant of the rover exploration problem known as rover domain. A team of rovers are tasked with exploring a continuous two-dimensional space and observing points of interest (POI) that are either spread out uniformly or deployed strategically. Every POI has an associated value that is rewarded to the agents if they make a successful observation. An observation is successful if the rover observes the POI from within a certain radius of the POI, obs_r. The observation counts towards the system performance for the closest rover that first observed it. Therefore, multiple rovers exploring the same POI does not accumulate system reward. Rovers typically will also have a limited time to explore the region. In this setting, the optimal strategy is for rovers to spread out and observe as many unique POIs as possible.

Every rover is equipped with two sensors. The first sensor is capable of detecting POIs and the second sensor can detect other rovers in the system. To keep the input state space of the rovers independent of the number of POIs and rovers in the system, the sensors collect data as densities in the region of operation. The sensors have a fixed resolution that defines the granularity of their density observations. Keeping the state space constant is important to support a robust learning architecture and facilitates addition and removal of rovers from the system during the course of exploration. The input to the density sensors are represented by Eqns. 5 and 6.

$$S_{rover,s} = \sum_{j \in J_s} \frac{1}{d(i,j)} \tag{5}$$

$$S_{poi,s} = \sum_{k \in K_s} \frac{v_k}{d(i,k)} \tag{6}$$

In equation (5), s is the quadrant, d measures the euclidean distance between the sensing rover i and other rover j; J_s is the set of all rovers in quadrant s. In equation (6), d measure the euclidean distance between the sensing rover i and the POI k; K_s is the set of POIs in the quadrant s.

The tightly-coupled version of the rover domain adds a coordination constraint to every POI. To make a successful observation, multiple agents must simultaneously observe the POI. The closest rovers that contribute to the coordination constraint are successful. Like before, only the first unique successful observation contributes to the system performance. In this setting, the optimal strategy for a uniformly distributed POI environment is for rovers to form teams and spread out. The system reward is computed using Eqn 7.

$$G(z) = \sum_k \frac{\prod_i N_{(i,k)} V_k}{\frac{1}{n}\sum_j d(i,k)} \tag{7}$$

Where, $G(z)$ is the system reward for z, the joint state-action of the rovers, V_k is the value of the POI k, $N_{i,k}$ is an indicator function that is true if the rover i is within the observation radius of the POI k and $d(i,k)$ is the euclidean distance between rover i and POI k.

4 REWARD APPROXIMATION AS A GAUSSIAN PROCESS

To create stepping stone rewards by evaluating the impact of adding counterfactual agents, D_{++} needs access to the functional form of the system reward. This hinders its application to most multiagent systems where the system reward is either unavailable or computationally expensive. A potential solution is to generate stepping stone rewards using an approximated system reward. Function approximators like neural networks can be used to approximate the system

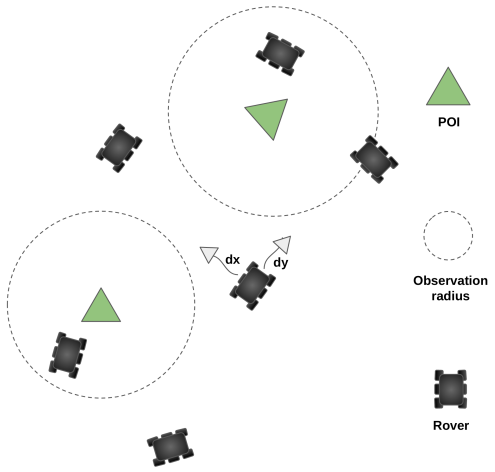


Figure 1: Rovers in the coordination problem must simultaneously make observations of the POI from within its observation radius. Every rover has sensors to detect the POI and rover density around it.

reward if they have sufficient training data. For tightly-coupled multiagent systems where agents are not likely to stumble upon good joint actions, the data at potentially valuable joint actions is sparse and therefore hard to learn. To account for the sparsity in the data, we need a probabilistic mechanism that can provide uncertainty estimates in the absence of observed data [9]. A Gaussian Process is a probabilistic function approximation method that provides confidence in its estimates. When used to approximate the system reward, a Gaussian Process can generate a distribution over the future rewards by assigning a probability to the potentially infinitely many functions that can fit the current observations [16, 21]. The mean of this probability distribution then represents the most probable characterization of the observations. Furthermore, using a probabilistic approach allows us to account for uncertainty in the estimated mean. For a set of a random variables $f(x)$ and input X, the joint distribution of these is also Gaussian.

$$p(f|X) = \mathcal{N}(f|\mu, K) \tag{8}$$

where, $f = (f(x_1, ..., x_n))$ is the set of random variables, $\mu = (\mu(x_1), ..., \mu(x_n))$ are the means (usually set to $\mu = 0$) and $K = k(x_a, x_b)$ is a positive definite covariance function. The kernel K measures similarity between points and allows injecting prior information about the properties of the model.

After observing some rewards r, the posterior can be used for estimating f_*, given new input X_* using Eqn. 9, 10.

$$p(f_*|X_*, X, r) = \int p(f_*|X_*, f)p(f|X, r)df \tag{9}$$

$$p(f_*|X_*, X, r) = \mathcal{N}(f_*|\mu_*, \Sigma_*) \tag{10}$$

4.1 Exploratory Gaussian Reward

Exploratory Gaussian Reward (EGR) models the system reward over every state dimension as a random variable. Initially, the uncertainty in reward estimates is uniformly high for the entire state space. As agents start exploring the state, the corresponding regions in the Gaussian Process start to fit to the true system reward. When agents are closer to POIs, to generate stepping stone rewards, agents use the expected variance in the reward estimate after adding hypothetical counterfactual partners. Because the expected variance is high for unobserved states, it is an optimistic estimate of the potential reward in those states. After sufficient exploration, the variance in estimates decreases and the optimistic stepping stones are reduced to their true potential. Even for highly undesired states, the optimistic rewards will initially still be positive. The agents will avoid these regions after only a few observations. By capturing the joint distribution over future rewards rather than their expectation, agents can focus on exploring states with higher variance that might be potentially valuable.

4.2 D_{++} with Exploratory Gaussian Reward

EGR works within the existing D_{++} framework. Unlike classical D_{++} that requires the system reward $G(z)$ to evaluate the effect of adding counterfactual partners, EGR uses an approximated $\hat{G}(z)$ that is sampled from the Gaussian Process. Initially, when the agents have not explored the state space enough, the sampled $\hat{G}(z)$ provides no feedback of the system performance (with the initial mean $\mu = 0$, $\hat{G}(z)$ gives a reward of zero). At every time step when agents receive the true system reward $G(z)$, the Gaussian posterior is updated. To compute stepping stone rewards, agents add counterfactual partners near POIs and sample the mean μ_* and variance Σ_* at this joint action. If the joint action was previously explored by the agent, the mean μ_* is close to the true value of the system reward with a variance representing the confidence of the estimate. On the other hand, an unobserved joint action will have a mean μ_* which is an estimate of the system reward based on the mean around the sampled joint action and a high variance. In such cases, the high variance provides an optimistic stepping stone reward that pushes the agent to coordinate in this joint action region.

The D and D_{++} rewards are computed using Eqn. 11 through 13.

$$\hat{D}(z) = G(z) - \hat{G}(z_{-1}) \tag{11}$$

$$D_{++}(i, n, z) = \frac{\hat{G}(z_{+(\cup_{i=1,...,n})}) + \alpha\hat{G_v}(z_{+(\cup_{i=1,...,n})}) - G(z)}{n} \tag{12}$$

$$\hat{G_v}(z_{+(\cup_{i=1,...,n})}) = \sqrt{(\Sigma_*)} \tag{13}$$

Where, $\hat{G}(z)$ is the reward estimate of the Gaussian Process, $\hat{G_v}$ is the standard deviation, and α is a constant factor to control the level of optimism. In practise, the uncertainty is clipped to a range between 0.4 and 1.2 when used to create stepping stone rewards to provide stability in estimates. This can also be achieved by normalizing the uncertainty. Algorithm 1 outlines the modified D_{++} algorithm that uses EGR for creating stepping stone rewards.

Algorithm 1: Optimistic Stepping Stones with GP approximation

Update the posterior $\hat{G}(z)$ based on observation $G(z)$
Calculate $\hat{D}_i(z)$ using Eqn. 11
Calculate $\hat{D}_{++}(N_A - 1)$ using Eqn. 12
if $\hat{D}_{++}(N_A - 1) \leq \hat{D}_i(z)$ **then**
  return $\hat{D}_i(z)$
else
  $n = 0$
  while $n < N_A - 1$ **do**
    $n = n + 1$
    break if $\hat{G_v}(n) <$ **threshold**
    Calculate $\hat{D}_{++}(n)$ using Eqn. 12
    if $\hat{D}_{++}(n) > \hat{D}_{++}(n-1)$ **then**
      Return $\hat{D}_{++}(n)$
    end if
  end while
end if
Return $\hat{D}_i(z)$

5 EXPERIMENTAL SETUP

We carry out several experiments in the tightly coupled rover coordination problem to investigate the performance and behavior of EGR. For all the experiments, the input state representation of the environment is a vector of size $(2 * 360\gamma)$, where γ is the resolution of the rover sensors. We use a resolution of 90° that gives rovers 4 density values for POIs and 4 for other rovers in the environment. Every agent uses a fully connected feed-forward neural network that maps the input state vector to a corresponding (dx, dy) action pair that lies in $[-1, 1]^2$. At every step of the simulation, agents take an action (dx, dy) based on their current policy that decides their movement in the x and y directions. The weights of the neural network are updated using policy gradient methods. Every agent also maintains a local Gaussian Process to approximate the system reward. The posterior approximation can be computed and updated locally by all agents and does not require access to the global state or the functional form of the reward. To account for the continuous state space and smooth reward values around POIs, we use a squared exponential kernel as the covariance function for the Gaussian Process. The covariance will control the shape of our estimated distribution and ultimately determine the characteristics of the system reward function. we assume the mean of the function to be zero and use an appropriate variance that allows the Gaussian Process to sample functions within the reward constraints of the environment.

6 RESULTS AND DISCUSSION

For all experiments, we compare the performance of EGR with the neural network approximation baseline and an unrealistic upper bound scenario which has access to the functional form of the system reward. Fig. 2 shows the two distinct coordination tasks performed in the rover coordination problem.

6.1 Uniform POI distribution

In the first experiment, we gauge the performance of EGR against the unrealistic upper bound by setting up an environment that captures the full complexity of the tightly coupled coordination problem. The environment is 30 x 30 units, with 18 POIs and 10 rovers. Rovers have to coordinate and observer multiple POIs to maximise the system performance. The POIs and the agents are randomly placed within the environment. The observation radius is bound between [2, 5]. The varied observation radius adds complexity to the task by inducing pressure to explore and observe POIs that are more accessible. All POIs have a coupling requirement of three.

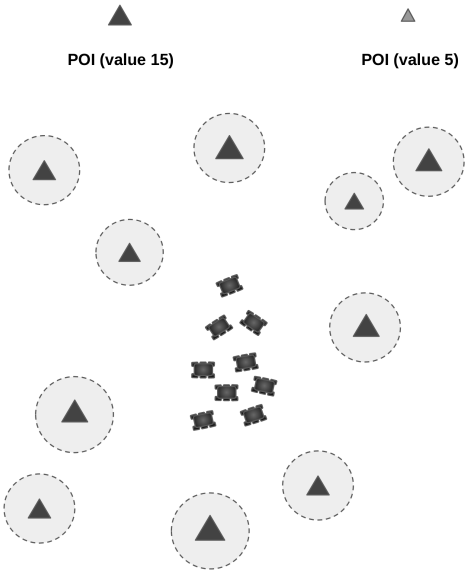
Fig. 2a shows the configuration of the environment and Fig. 3 shows the typical trajectories of rovers after learning to coordinate using EGR and D_{++}. Fig. 4 shows the performance of EGR compared to the approximation baseline. Performance of agents with complete access to the functional form of the system reward is also plotted. This is an unrealistic upper bound as most systems will not have a computable and noiseless functional form of the reward.

The baseline that approximates system reward with a neural network does not perform well as the stepping stone reward estimates are roughly uniform throughout the state space. Because agents are very unlikely to bump into coordination states, the neural network does not make enough observations to learn to predict valuable stepping stone rewards. EGR outperforms the baseline by a wide margin. The overall performance of EGR is also sufficiently comparable to the unrealistic upper bound given that EGR depends completely on local agent information and is free from the functional form the of the system reward.

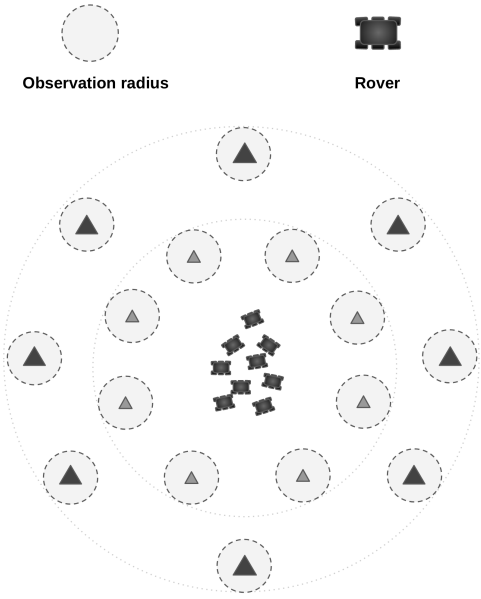
6.2 Concentric POI configuration

The second experiment is designed to highlight the capability of EGR for optimistic exploration. Two sets of POIs are placed in the environment in concentric circles. Fig. 2b shows this configuration. The inner ring of POIs have a coupling of 3 and provide a reward of 5 if the coupling is satisfied. The POIs in the outer ring are more valuable and provide a reward of 15 for a coupling of 3. Agents are placed at the center of the environment inside the inner ring. The maximum number of timesteps per episode is adjusted to make sure agents would only have sufficient time to coordinate and observe about 40% of the total POIs. Agents would need to explicitly seek out the POIs in the outer ring to maximize the system performance.

This experiment tests the ability of EGR to push agents to explore potentially valuable unknown regions of the state space. Fig. 7 compares the performance of EGR with classic D_{++}. This result highlights the benefits of exploration in a tightly coupled problem. Without the explicit need of exploring, agents that use D_{++} learn to coordinate and observe the POIs in the inner ring. Removing an agent from this stable learned configuration decreases the system performance, thus reinforcing the agents to continue to exploit the rewards in the inner ring. Initially, agents using EGR also learn to coordinate and observe POIs in the inner ring. The state space region beyond the inner ring is unexplored and agents are pushed into exploring it as the positive optimistic stepping stone rewards are sufficient to balance the negative effect on system performance incurred due to an agent leaving the inner ring. Fig. 5 shows the



(a) Experimental setup 1. POIs are uniformly distributed in the world. Every POI has the same coupling requirement and reward for observation.



(b) Experimental setup 2. POIs are distributed in 2 concentric rings with the higer values POIs in the outer ring. Every POI has the same coupling requirement and observation radius.

Figure 2: Experiments are performed on 2 setting of the rover domain to highlight distinct coordination tasks.

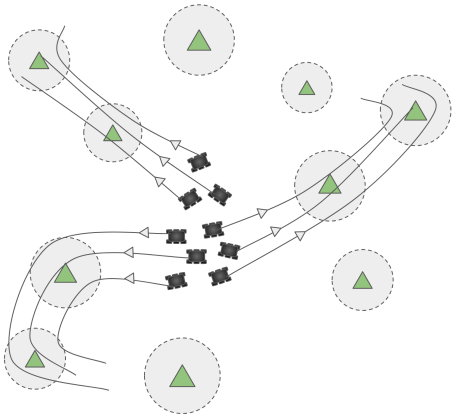


Figure 3: Uniform POI distribution: In this setting, the optimal strategy for the rovers is to form teams and explore.

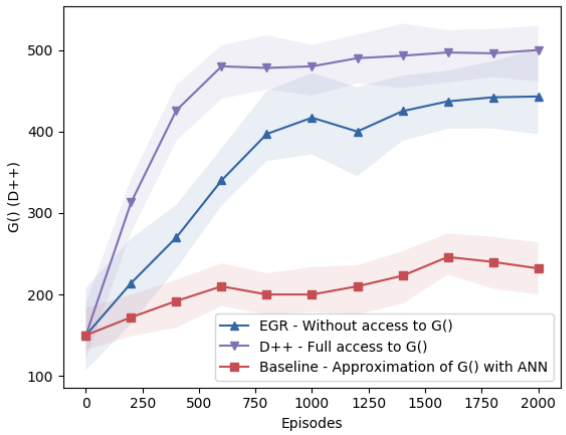
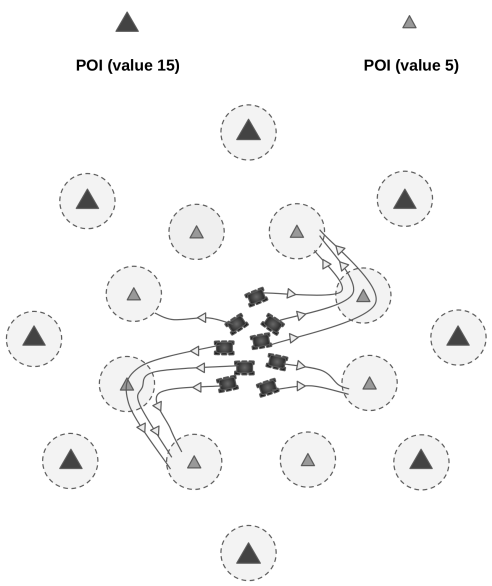
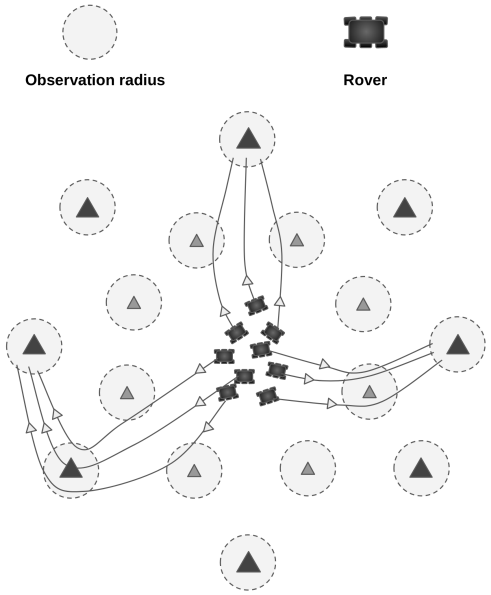


Figure 4: Uniform POI distribution: Comparing performance of reward models in tightly coupled rover domain. There are 18 rovers and 10 POIs in a 30x30 world. EGR significantly outperforms the neural network reward approximation baseline.

behavior of rovers using D_{++} and EGR in this setting. After sufficient episodes, agents using EGR eventually learn to disregard the inner ring and instead team up to coordinate and observe the higher value POIs in the outer ring. Although EGR takes longer to converge, it outperforms classic D_{++} without access to the global system state and access to a noiseless functional form of the system performance. Similarly, Fig. 6 shows the behavior of rovers using EGR when the inner ring has higher value POIs. Although the outer ring is a lower value region, rovers are still pushed to explore it due to the uncertainty in their reward estimates. After observing the true rewards in the outer ring, rovers return to the inner ring.



(a) The figure shows trajectories for rovers using D_{++}. Breaking a team and exploring other regions reduces the system performance and is negatively reinforced; Agents get stuck in a local optimum.



(b) The figure shows trajectories for rovers using EGR. Although breaking a team and exploring other regions would reduce the system performance, the optimistic stepping stone rewards generated by EGR negate this effect and push rovers out of the local optimum.

Figure 5: Concentric POI configuration: POIs are distributed in 2 concentric rings with the higher value POIs in the outer ring. Agents start from the middle of the inner ring. POIs in the inner ring are easier to get to and agents learn to observe them by forming teams.

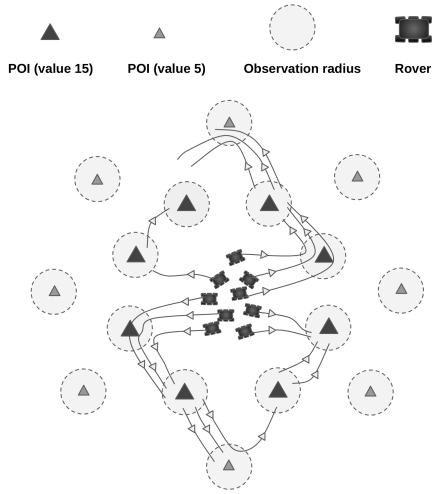


Figure 6: Concentric POI distribution: The figure shows trajectories for rovers using EGR. Although the optimistic stepping stone rewards generated by EGR push rovers to the outer low value region, after adequate observations, rovers return to the inner ring.

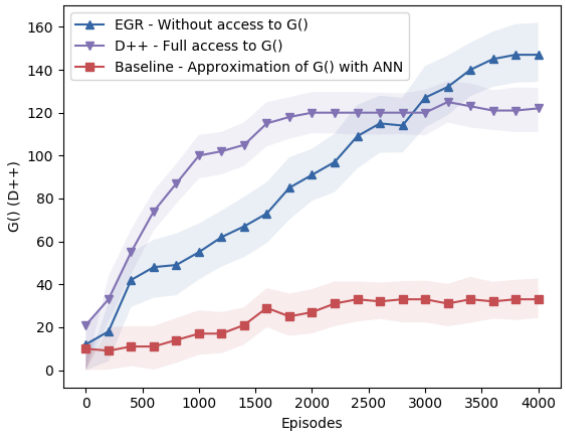


Figure 7: Concentric POI configuration: Comparing performance of reward models in tightly coupled rover domain. POIs are distributed in 2 concentric rings. Agents start from the middle of the inner ring.

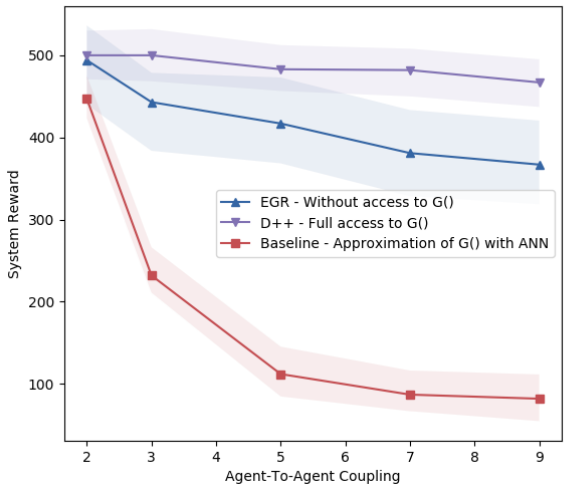


Figure 8: Comparing performance of reward models for tight coupling constraints.

6.3 EGR with tighter coordination constraints

In the third experiment, we investigate the robustness of EGR by scaling the coordination constraints. Tighter coupling constraints further increase the complexity of the problem by reducing the likelihood of agents bumping into POIs together. The number of POIs in the environment is scaled down as the constraints are tightened to make sure agents have sufficient time to explore and coordinate.

The experiment is designed to test the robustness of EGR for high coordination requirements. Fig. 8 compares the performance of EGR with the baseline. For a coupling of 2, the performance of EGR is comparable to the baseline and D_{++} with access to the system reward. This is a loose coordination requirement that can be fulfilled with rewards generated using D. Beyond a coupling of 2, the baseline fails to learn as the likelihood of agents stumbling upon coordination states decreases with higher coupling. EGR performs comparable to agents with full access to global reward and performance only slightly degrades for higher coupling constraints.

7 CONCLUSIONS

In this work, we introduced a new method of creating optimistic stepping stone rewards, EGR, which allows agents in high coordination requirement problems to find suitable partners for coordination. By leveraging the uncertainty estimates of a Gaussian Process to model system reward in EGR, we see that agents learn to coordinate using D_{++} without access to the functional form of the system reward. EGR also allows for intuitive injection of prior knowledge to further shape rewards and speed-up the learning process. We show that EGR outperforms the baseline by a wide margin and has a comparable performance to agents which have full access to the global state and functional form of the system reward. Finally, the results demonstrate the impressive scalability of EGR which maintained superior performance as the agent-to-agent coordination coupling in the system was increased (coupling of up to 9) while the performance of the baseline degraded rapidly.

EGR assumes uniform reward uncertainty over potential coordination states for all agents in the system. In future work, we will explore how EGR could be extended when the system has regions of highly varying reward. In systems where certain state space regions are more valuable than others, it would be necessary to be able to bias exploration towards those regions to speed up learning. Additionally, we will explore how EGR performs in problems where diverse agents with different action spaces must coordinate with other agents or human partners.

8 ACKNOWLEDGEMENT

This work was partially supported by the Air Force Office of Scientific Research under grant No. FA9550-19-1-0195 and the National Science Foundation under grant No. IIS-1815886.

REFERENCES

[1] A. Agogino and K. Tumer. 2008. Efficient evaluation functions for evolving coordination. *Evolutionary Computation* 16, 2 (2008), 257–288. https://doi.org/10.1162/evco.2008.16.2.257

[2] Adrian K Agogino and Kagan Tumer. 2008. Analyzing and visualizing multiagent rewards in dynamic and stochastic domains. 17, 2 (2008), 320–338.

[3] Itamar Arel, Cong Liu, Tom Urbanik, and Airton G Kohls. 2010. Reinforcement learning-based multi-agent system for network traffic signal control. *IET Intelligent Transport Systems* 4, 2 (2010), 128–135.

[4] Ronald C Arkin and Jonathan Diaz. 2002. Line-of-sight constrained exploration for reactive multiagent robotic teams. In *7th International Workshop on Advanced Motion Control. Proceedings (Cat. No. 02TH8623)*. IEEE, 455–461.

[5] Monica Babes, Enrique Munoz De Cote, and Michael L Littman. 2008. Social reward shaping in the prisoner's dilemma. In *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems-Volume 3*. International Foundation for Autonomous Agents and Multiagent Systems, 1389–1392.

[6] Michael Bowling and Manuela Veloso. 2003. Simultaneous adversarial multi-robot learning. In *IJCAI*, Vol. 3. 699–704.

[7] Lucian Bu, Robert Babu, Bart De Schutter, et al. 2008. A comprehensive survey of multiagent reinforcement learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 38, 2 (2008), 156–172.

[8] Yongcan Cao, Wenwu Yu, Wei Ren, and Guanrong Chen. 2012. An overview of recent progress in the study of distributed multi-agent coordination. *IEEE Transactions on Industrial informatics* 9, 1 (2012), 427–438.

[9] Georgios Chalkiadakis and Craig Boutilier. 2003. Coordination in multiagent reinforcement learning: a Bayesian approach. In *Proceedings of the second international joint conference on Autonomous agents and multiagent systems*. ACM, 709–716.

[10] M. Colby, T. Duchow-Pressley, J. J. Chung, and K. Tumer. 2016. Local Approximation of Difference Evaluation Functions. In *Proceedings of the Fifteenth International Joint Conference on Autonomous Agents and Multiagent Systems*. Singapore, 521–529.

[11] Mitchell Colby, Logan Yliniemi, and Kagan Tumer. 2016. Autonomous multiagent space exploration with high-level human feedback. *Journal of Aerospace Information Systems* (2016), 301–315.

[12] Sylvain Damiani, Gérard Verfaillie, and Marie-Claire Charmeau. 2005. An earth watching satellite constellation: How to manage a team of watching agents with limited communications. In *Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*. ACM, 455–462.

[13] Sam Devlin and Daniel Kudenko. 2011. Theoretical considerations of potential-based reward shaping for multi-agent systems. In *The 10th International Conference on Autonomous Agents and Multiagent Systems-Volume 1*. International Foundation for Autonomous Agents and Multiagent Systems, 225–232.

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

[15] Kurt Dresner and Peter Stone. 2005. Multiagent Traffic Management: An Improved Intersection Control Mechanism. In *Proceedings of the Fourth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS '05)*. ACM, New York, NY, USA, 471–477. https://doi.org/10.1145/1082473.1082545

[16] Yaakov Engel, Shie Mannor, and Ron Meir. 2005. Reinforcement learning with Gaussian processes. In *Proceedings of the 22nd international conference on Machine learning*. ACM, 201–208.

[17] Ettore Ferranti, Niki Trigoni, and Mark Levene. 2007. Brick& Mortar: an online multi-agent exploration algorithm. In *Proceedings 2007 IEEE International Conference on Robotics and Automation*. IEEE, 761–767.

[18] Jakob N Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. 2018. Counterfactual multi-agent policy gradients. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

[19] Jared Hill, James Archibald, Wynn Stirling, and Richard Frost. 2005. A multi-agent system architecture for distributed air traffic control. In *AIAA guidance, navigation, and control conference and exhibit*. 6049.

[20] Pieter Jan't Hoen, Karl Tuyls, Liviu Panait, Sean Luke, and J.A. La Poutré. 2005. An Overview of Cooperative and Competitive Multiagent Learning. *Learning and Adaption in Multi-Agent Systems* (2005), 1–50.

[21] Malte Kuss and Carl E Rasmussen. 2004. Gaussian processes in reinforcement learning. In *Advances in neural information processing systems*. 751–758.

[22] Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, Pieter Abbeel, and Igor Mordatch. 2017. Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments. *CoRR* abs/1706.02275 (2017). arXiv:1706.02275 http://arxiv.org/abs/1706.02275

[23] Andrew Y. Ng, Daishi Harada, and Stuart J. Russell. 1999. Policy Invariance Under Reward Transformations: Theory and Application to Reward Shaping. In *Proceedings of the Sixteenth International Conference on Machine Learning (ICML '99)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 278–287. http://dl.acm.org/citation.cfm?id=645528.657613

[24] Liviu Panait and Sean Luke. 2005. Cooperative multi-agent learning: The state of the art. *Autonomous agents and multi-agent systems* 11, 3 (2005), 387–434.

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

[26] Peter Stone and Manuela Veloso. 2000. Multiagent systems: A survey from a machine learning perspective. *Autonomous Robots* 8, 3 (2000), 345–383.

[27] Ming Tan. 1993. Multi-Agent Reinforcement Learning: Independent vs. Cooperative Agents. In *In Proceedings of the Tenth International Conference on Machine Learning*. Morgan Kaufmann, 330–337.

[28] Claire Tomlin, George J Pappas, and Shankar Sastry. 1998. Conflict resolution for air traffic management: A study in multiagent hybrid systems. *IEEE Transactions on automatic control* 43, 4 (1998), 509–521.

[29] Kagan Tumer and Adrian Agogino. 2007. Distributed agent-based air traffic flow management. *Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems - AAMAS '07* (2007), 1. https://doi.org/10.1145/1329125.1329434

[30] Karl Tuyls and Kagan Tumer. 2016. Multiagent Learning. In *Multiagent Systems* (2nd ed.), Gerhard Weiss (Ed.). The MIT Press, London, England, Chapter 10, 423–484.

[31] Jijun Wang, Michael Lewis, and Paul Scerri. 2006. Cooperating robots for search and rescue. In *Proceedings of the AAMAS 1st International Workshop on Agent Technology for Disaster Management*. 92–99.

[32] Logan Yliniemi and Kagan Tumer. 2014. Multi-objective multiagent credit assignment through difference rewards in reinforcement learning. In *Asia-Pacific Conference on Simulated Evolution and Learning*. Springer, Springer, 407–418.