# Graphical Models in Continuous Domains for Multiagent Reinforcement Learning

# (Extended Abstract)

Scott Proper
Oregon State University
Corvallis, OR 97331, USA
proper@eecs.oregonstate.edu

Kagan Tumer
Oregon State University
Corvallis, OR 97331, USA
kagan.tumer@oregonstate.edu

## ABSTRACT

In this paper we test two coordination methods – difference rewards and coordination graphs – in a continuous, multiagent rover domain using reinforcement learning, and discuss the situations in which each of these methods perform better alone or together, and why. We also contribute a novel method of applying coordination graphs in a continuous domain by taking advantage of the wire-fitting approach used to handle continuous state and action spaces.

## Categories and Subject Descriptors

H.3.4 [**Systems and Software**]: Distributed Systems

## General Terms

Algorithms, Performance, Experimentation

## Keywords

Multiagent Coordination, Reinforcement Learning

## 1. INTRODUCTION

The problem of scaling reinforcement learning to large multiagent domains is very challenging because the agents in the system provide a constantly changing environment in which each agent needs to learn its task. Agents must somehow learn to coordinate among themselves and develop a joint set of policies to solve the problem.

In this paper, we examine two popular forms of coordination: coordination graphs, which allow small groups of agents to coordinate actions based domain-specific dependencies, represented by a graph of edges between agents [4]; and difference rewards, which are a specific type of shaped reward that encourages good agent behavior by rewarding actions that are closely aligned with the desired overall system behavior, while still allowing agents to learn from the reinforcement signal [1]. We compare these techniques in a simulated domain of continuous rover navigation and observation with multiple agents. Our conclusions should be helpful for future researchers who wish to understand the differences between these methods for their own applications. The results illustrate some of the possible differences between domains that can cause one or another coordination method to be an appropriate choice.

**Appears in:** *Proceedings of the 12th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2013)*, Ito, Jonker, Gini, and Shehory (eds.), May, 6–10, 2013, Saint Paul, Minnesota, USA.

## 2. BACKGROUND

We combined several methods for coordination and learning in this paper: Q-learning, wire-fitting, advantage learning, coordination graphs, and difference rewards. We discuss these briefly below.

**Q-learning:** Q-learning is a well-known reinforcement learning algorithm for solving Markov Decision Processes (MDP). We adapt a heavily-modified version of this algorithm in this paper.

**Wire-fitting:** To deal with the problem of acting in a space in which states and actions may be continuous, Baird and Klopf propose the wire fitting algorithm, which efficiently stores an approximation of a continuous Q-function [2]. They propose using a function approximator (in this paper, a neural network) to concurrently output a fixed number of actions and corresponding values ("wires"), given a certain state. We use this wire-fitting approach in this paper to deal with the continuous nature of the rover domain.

**Difference rewards:** When providing the reward signal to an agent, one option is to provide each agent with the **global reward** $G(z)$ for each step. However, this reward signal is insensitive to an agent's actions and for large systems, leads to particularly slow learning. Instead, we can apply a reward shaping method known as difference rewards, which are given by $D_i(z) = G(z) - G(z - z_i)$, where $z - z_i$ specifies the state of the system without agent $i$ [1]. The difference reward has two key advantages: first, it provides an agent with a "cleaner" signal than $G$. Second, because the second term does not depend on the actions of agent $i$, any action by agent $i$ that improves $D$, also improves $G$.

## 3. COORDINATION GRAPHS

A coordination graph (CG) can be described over a system of agents to represent the coordination requirements of that system. A CG contains a node for each agent and an edge between pairs of agents if they must directly coordinate their actions to optimize the global policy. Typically, the assumption made for CGs is that the global reward may be decomposed as a sum $R = \sum_i r_i$ of local agent rewards. In this paper, we allow a variety of possible ways in which the reward may be decomposed. As in Kok and Vlassis [4] our implementation is context-specific, thus we use an edge-based decomposition with an agent-based update. We combine this with the Max-plus algorithm [4] for action selection.

Several issues arise when adapting CGs for use in continuous domains. One issue is the use of advantage learning [3]. We derive the agent-based update equation using the advantage learning update as:

$$Q_{ij}(\mathbf{s}_{ij}, a_i, a_j) \xleftarrow{\alpha} \sum_{b \in \{i,j\}} \frac{\frac{1}{k}\left[r_b(\mathbf{s}, \mathbf{a}) + \gamma Q_b(\mathbf{s}'_b, a^*_b)\right] + (1 - \frac{1}{k}) Q_b(\mathbf{s}_b, a^*_b)}{|\Gamma(b)|} \quad (1)$$

This update propagates the temporal-difference error from all edges including agents $i$ and $j$ to the Q-function of each edge $(i,j)$.

A second issue is that it is not always clear from the dynamics of the domain which agents should coordinate with each other. Ideally, all agents should coordinate, but this is impractical. We choose to limit coordination based on two factors: distance, and an additional upper limit on the number of neighbors for each agent. If there are too many potential neighbors within the distance limit, the closest neighbors are chosen. This forces the CG into a limited coverage allowing practical application of Max-Plus algorithm to compute a good joint action in reasonable time.

A further problem is that the Max-Plus algorithm requires a finite number of actions that each agent may select from. Unfortunately, a continuous action space has an infinite number of possible actions. Thus, we select a set of "candidate" actions for each agent from the output of the wire-fitter storing the value function for each agent. Only the top three candidates are selected.

## 4. CONTINUOUS ROVER PROBLEM

We implement a variation of the continuous rover problem, as described in [1]. In this problem, there is a set of rovers on a two dimensional plane which are trying to observe points of interests (POIs). A POI has a fixed position on the plane and has a value associated with it. At every time step, the rovers sense the world through eight continuous sensors. From a rover's point of view, the world is divided up into four quadrants, with two sensors per quadrant. For each quadrant, the two corresponding sensors each return a function of the POIs or rovers in that quadrant at that moment [1]. These eight sensor inputs provide the inputs to a neural network which stores the policy for each rover.

We require $N$ rovers to observe a POI simultaneously in order to receive a reward. We do this by defining a maximum observation distance $\delta_0$. The global, local, and difference rewards $G$, $L$, and $D$ for the rover problem are as follows:

$$G(z) = \sum_i \frac{V_i \cdot N \cdot I(|c_i| \geq N)}{\sum_{k=1}^{N} \delta_{i,c_i(k)}}$$

$$L_j(z) = \sum_i \frac{V_i \cdot I(\delta_{i,j} \leq \delta_0) \cdot I(|c_i| \geq N)}{\delta_{i,j}}$$

$$D_j(z) = \sum_i \begin{cases} \frac{V_i N}{\sum_{k=1}^{N} \delta_{i,c_i(k)}} - \frac{V_i N}{\sum_{k=1,k\neq j}^{N+1} \delta_{i,c_i(k)}} & \text{if } |c_i| > N \wedge j \in \{c_i(1)...c_i(N)\} \\ \frac{V_i N}{\sum_{k=1}^{N} \delta_{i,c_i(k)}} & \text{if } |c_i| = N \wedge j \in c_i \\ 0 & \text{otherwise} \end{cases}$$

where $V_i$ is the value of the $i$th POI, $c_i$ is the set of the agents within viewing range $\delta_0$ of POI $i$ and $c_i(k)$ is the $k$th closest agent, $N$ is the minimum number of agents required to make a successful observation, and $I(|c_i| \geq N)$ and $I(\delta_{i,j} \leq \delta_0)$ are indicator functions. The observation information $\delta_{x,y} = max\{\|x-y\|^2, d^2\}$ results from observing a POI and is bounded by a minimum observation distance $d$ [1].

## 5. EXPERIMENTAL RESULTS

We conducted experiments with a 10-agent, 30-POI version of the Rover domain with a dynamic, randomly generated distribution of agents and POIs, averaged over 30 runs. All results were measured against the global reward $G(z)$. Error bars are calculated using the sample standard error of the mean $\sigma/\sqrt{30}$.

We compared experiments with four variations on the rover domain: two experiments with agents starting in very congested conditions, uniformly distributed in 5x5 area at the center position, and two experiments with the agents uniformly scattered in the same 70x70 area the POIs are scattered. These experiments were further varied based on the minimum number of agents required to perform an observation: $N = 1$ or $N = 3$. The result of only one of these experiments is shown in Figure 1. Here, the learning problem is for agents to join up across their scattered positions to observe a POI together. This is an uncongested domain, so difference rewards do not contribute to the solution. However, CGs are necessary for agents to find each other and navigate to a POI together; without them, agents do not learn to converge and observe POIs.

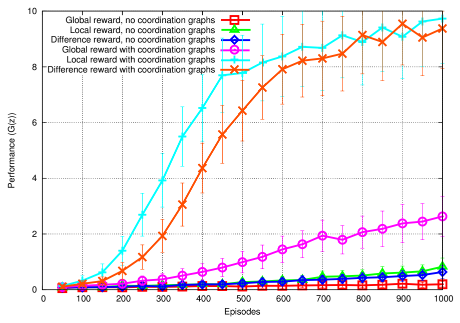

**Figure 1: Results with agents starting scattered, with $N = 3$.**

In our experiments with agents all starting near the center, we observed that difference rewards were very useful in relieving the congested conditions. However, if we require only one agent to make an observation (i.e. $N = 1$) the addition of CGs adds unnecessary complexity to the learner: CGs are not needed to solve a coordination problem in this domain and thus the extra parameters of the neural network required to store the CG only slow convergence.

From our results, we conclude that while CGs and $D$ solve related problems, the particular issues each method is suited to differ. Difference rewards are most useful in highly congested domains. CGs are well-suited to cases in which agent pairs must take specific coordinated actions together to succeed. Including either form of coordination when it is not needed may be harmful, but rarely prevents learning from taking place and may only slow convergence slightly. Cases exist when using both coordination techniques together is helpful, i.e. when a problem requires more than one form of coordination. These conclusions are supported by past work involving these methods: coordination graphs have primarily been used for domains in which tight coordination between agents is needed [4], while difference rewards have primarily been applied to domains in which congestion is an important factor [1].

**Acknowledgements:** This work was partially supported by NSF grant CNS-0931591 and NETL grant DE-FE0000857.

## 6. REFERENCES

[1] A. K. Agogino and K. Tumer. Analyzing and visualizing multiagent rewards in dynamic and stochastic environments. *Journal of Autonomous Agents and Multi Agent Systems*, 17(2):320–338, 2008.

[2] L. Baird and H. Klopf. Reinforcement learning with high-dimensional continuous actions. Technical Report WL-TR-93-1147, Wright Laboratory, Wright-Patterson Air Force Base, 1993.

[3] M. E. Harmon and L. C. Baird. Residual advantage learning applied to a differential game. In *In Proc. of the International Conference on Neural Networks, Washington D.C*, 1995.

[4] J. R. Kok and N. A. Vlassis. Collaborative multiagent reinforcement learning by payoff propagation. *J. Mach. Learn. Res.*, 7:1789–1828, 2006.