# Reinforcement Learning in Large Multi-agent Systems

Adrian Agogino
UC Santa Cruz, NASA Ames Research Center
Mailstop 269-3
Moffett Field, CA 94035, USA
adrian@email.arc.nasa.gov

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

## ABSTRACT

Enabling reinforcement learning to be effective in large-scale multi-agent Markov Decisions Problems is a challenging task. To address this problem we propose a multi-agent variant of Q-learning: "Q Updates with Immediate Counterfactual Rewards-learning" (QUICR-learning). Given a global reward function over all agents that the large-scale system is trying to maximize, QUICR-learning breaks down the global reward into many agent-specific rewards that have the following two properties: 1) agents maximizing their agent-specific rewards tend to maximize the global reward, 2) an agent's action has a large influence on its agent-specific reward, allowing it to learn quickly. Each agent then uses standard Q-learning type updates to form a policy to maximize the agent-specific rewards. Results on multi-agent grid-world problems over two topologies, show that QUICR-learning can be effective with hundreds of agents and can achieve up to 300% improvements in performance over both conventional and local Q-learning in the largest tested systems.

## 1. INTRODUCTION

Applying single-agent reinforcement learning algorithms to large-scale multi-agent systems is difficult. A large-scale system typically cannot use a single reinforcement learner since the state-space of this single learner would be prohibitively large. Furthermore, the single learner presents a single point of failure and is unsuitable for fundamentally distributed problems, such as satellite control where there are communication delays.

A more promising approach is to give each agent in the large-scale system its own reinforcement learner. The main issue with this approach is how to entice the distributed collection of learners to form policies that maximize a global reward for the entire large-scale system. It is difficult for a single agent's learner to directly maximize the global reward since so many other agents influence this reward. In Markov Decisions Problems problems presented in this paper the global reward may be influenced by as many as 25,200 actions (actions of 400 agents over 63 time steps). Reinforcement learners that use rewards that are more local to a particular agent are also problematic, since often the policies formed by these learners maximize their local rewards while reducing the global reward. As an alternative, we present "Q Updates with Immediate Counterfactual Rewards learning" (QUICR-learning), which uses agent-specific rewards that suppress the impact of other agents. Rewards in QUICR-learning are both heavily agent-sensitive, making the learning task easier and aligned with the system level goal, ensuring that agents receiving high rewards are helping the system as a whole. These agent-specific reward functions are then used with standard temporal difference methods to create a learning method that is significantly faster than standard Q-learning in large multi-agent systems.

Most current multi-agent reinforcement learning methods are designed to work in domains with a moderate to small number of agents such as robotic soccer, multi-agent foraging and multi-agent grid-worlds[6, 3, 2]. Large numbers of agents are often used in ant colony algorithms [1] that solve the coordination problem by utilizing "ant trails," providing good results in path-finding domains. In [5] coordination between one hundred robots is achieved through the use of hierarchical dispatching and with spatial reasoning through the use of topological graphs.

In this paper, we present QUICR-learning which provides fast convergence in multi-agent learning domains. This reinforcement learning method is based on decomposing the global reward into agent-specific rewards, without assuming that the full system reward is linearly separable or requiring hand tuning based on domain knowledge. In Section 2 we discuss the temporal and structural credit assignment problems in multi-agent systems, and describe the QUICR-learning algorithm. In Section 3 we present results on two variants of a multi-agent gridworld problem, showing that QUICR-learning performs up to 400% better than standard Q-learning in multi-agent problems.

## 2. REINFORCEMENT LEARNING IN LARGE SYSTEMS

The goal of a reinforcement learner is to maximize the rewards received for a sequence of actions. Starting from current time step $t$, the undiscounted sum of rewards till a final time step $T$ can be represented by:

$$R_t(s_t(a)) = \sum_{k=0}^{T-t} r_{t+k}(s_{t+k}(a)) \; . \qquad (1)$$


*AAMAS'05,* July 25-29, 2005, Utrecht, Netherlands.

where $a$ is a vector containing the actions of all agents at all time steps, $s_t(a)$ is the state function returning the state of *all* agents for for a single time step, and $r_t(s)$ is the single-time-step reward function, which is a function of the states of *all* of the agents.

This reward is a function of all of the previous actions of all of the agents. Every reward is a function of the states of all the agents, and every state is a function of all the actions that preceded it (even though it is Markovian, the previous states ultimately depend on previous actions). In a system with $n$ agents, on average $\frac{1}{2}n * T$ actions affect reward. Agents need to use this reward to evaluate their single action, yet even in the idealized domains presented Section 3, with four hundred agents and sixty-three time steps there are an average of 12600 actions affecting the reward!

## 2.1 Standard Q-Learning

Reinforcement learners such as Q-learning address how to assign credit of future rewards to an agent's current action. The goal of Q-learning is to create a policy that maximizes the sum of future rewards, $R_t(s_t(a))$, from the current state [4, 7, 8]. It does this by maintaining tables of Q-values, which estimate the expected sum of future rewards for a particular action in a particular state. In the TD(0) version of Q-learning, a Q-value, $Q(s_t, a_t)$, is updated with the following Q-learning rule [1]:

$$\Delta Q(s_t, a_t) = \alpha(r_t + max_a Q(s_{t+1}, a)) \; . \tag{2}$$

The assumption with this update is that the action $a_t$ is most responsible for the immediate reward $r_t$, but is less responsible for the sum of future rewards, $\sum_{k=1}^{T-t} r_{t+k}(s_{t+k}(a))$. This assumption is reasonable since rewards in the future are affected by uncertain future actions and noise in state transitions. Instead of using the sum of future rewards directly to update its table, Q-learning uses a Q-value from the next state entered as an estimate for those future rewards. Under benign assumptions, Q-values are shown to converge to the actual value of the future rewards [8].

Eventhough Q-learning addresses how to account for rewards taken at other time steps, it does not address how to handle rewards that are a function of the actions of many agents. In standard Q-learning an agent will get full credit for actions taken by all of the other agents. As a result when they are many agents, standard Q-learning is generally slow since it will take many episodes for an agent to figure out its impact on a reward it barely influences.

## 2.2 Local Q-Learning

One way to address the problem of having a global reward that is a function of many agents and allow for fast learning is to assume that agents' actions are independent. Without this assumption, the immediate reward function for a multi-agent reward system may be a function of all the states:

$$r_t(s_{t,1}(a_1), s_{t,2}(a_1), \ldots, s_{t,n}(a_n)) \; ,$$

where $s_{t,i}(a_i)$ is the state for agent $i$ and is a function of only agent $i$'s previous actions. The number of states determining the reward grows linearly with the number of agents, while the number of actions that determine each state grows linearly with the number of time steps. To reduce the huge number of actions that affect this reward, often the reward is assumed to be linearly separable:

$$r_t(s_t) = \sum_i w_i r_{t,i}(s_{t,i}(a_i)) \; .$$

Then each agent receives a reward $r_{t,i}$ which is only a function of its action. Q-learning is then used to resolve the remaining temporal problem of how to use rewards received at other time steps. If the agents are actually independent, this method leads to a significant speedup in learning as an agent receives direct credit for its actions. If the agents are coupled, then the independence assumption still allows fast learning, but the agents will tend to converge to the wrong policy. With loose coupling the benefits of the assumption may still outweigh the costs when there are many agents. However, when agents are tightly coupled, the independence assumption may lead to unacceptable solutions and may even converge to a solution that is worse than random [9].

## 2.3 QUICR-Learning

In this section we present QUICR-learning, a learning algorithm for multi-agent systems that does not assume that the system reward function is linearly separable. Instead it uses a mechanism for creating rewards that are a function of all of the agents, but still provide many of the benefits of hand-crafted rewards. Many hand-crafted multi-agent learning algorithms exploit detailed knowledge about a domain to provide agent rewards that allow the system to maximize a global reward. These rewards are designed to have two beneficial properties: they are "aligned" with the overall learning task and they have high "sensitivity" to the actions of the agent.

The first property of alignment means that when an agent maximizes its own reward it tends to maximize the overall system reward. Without this property, a large multi-agent system can lead to agents performing useless work, or worse, working at cross-purposes. Reward sensitivity means that an agent's reward is more sensitive to its own actions than to other agents actions. This property is important for agents to learn quickly.

QUICR-learning is based on providing agents with rewards that are both aligned with the system goals and sensitive to the agent's states. It aims to provide the benefits of hand-crafted algorithms without requiring detailed domain knowledge. In a task where the reward can be expressed as in Equation 1, let us introduce the difference reward (adapted from [9]) given by:

$$D_t^i(s_t(a)) \quad = \quad R_t(s_t(a)) - R_t(s_t(a - a_{t,i}))$$

where $a - a_{t,i}$ denotes a counterfactual state where agent $i$ has not taken the action it took in time step $t$ (e.g., the action of agent $i$ has been removed from the vector containing the actions of all the agents before the system state has been computed). Decomposing further, we obtain:

$$\begin{aligned} D_t^i(s_t(a)) &= \sum_{k=0}^{T-t} r_{t+k}(s_{t+k}(a)) - r_{t+k}(s_{t+k}(a - a_{t,i})) \\ &= \sum_{k=0}^{T-t} d_{t+k}(s_{t+k}(a), s_{t+k}(a - a_{t,i})) \; . \end{aligned} \tag{3}$$

[1] This paper uses undiscounted learning to simplify notation, but all the algorithms also apply to discounted learning as well.

where $d_t(s_1, s_2) = r_t(s_1) - r_t(s_2)$. (We introduce the single time step "difference" reward $d_t$ to keep the parallel between Equations 1 and 3). This reward is much more sensitive to an agent's action than $r_t$ since much of the effects of the other agents are subtracted out with the counterfactual [9]. Unfortunately in general $d_t(s_1, s_2)$ is non-Markovian since the second parameter may depend of previous states, making its use troublesome in a learning task involving both a temporal and structural credit assignment.

In order to overcome this shortcoming of Equation 3, let us make the following two assumptions:

1. The counterfactual $a - a_{t,i}$ action moves agent $i$ to an absorbing state, $s_b$ that is independent of of its current state.

2. The future state of agents other than agent $i$ are not affected by the actions of agent $i$.

The first assumption forces us to compute a counterfactual state that is not necessarily a minor modification to agent $i$'s current state. Therefore, differential function estimation techniques that rely on a small change in agent $i$'s (e.g., Taylor series expansion) state cannot be used. However, each agent's countefactual state is for itself (e.g, not computed for other agents) and a single time step (e.g., the countefactual states do not propagate through time). The second assumption holds in many multi-agent systems, since to reduce the state-space to manageable levels, agents often do not directly observe each other (though are still coupled through the reward).

Given these conditions, the counterfactual state for time $t + k$ is computed from the actual state at time $t + k$, by replacing the state of agent $i$ at time $t$ with $s_b$. Now the difference reward can be made into a Markovian function:

$$d_t^i(s_t) = r_t(s_t) - r_t(s_t - s_{t,i} + s_b) \ , \tag{4}$$

where the expression $s_t - s_{t,i} + s_b$ denotes replacing agent $i$'s state with state $s_b$.

Now the Q-learning rule can be applied to the difference reward, resulting in the QUICR-learning rule:

$$\begin{aligned} \Delta Q(s_t, a_t) &= \alpha(r_t(s_t) - r_t(s_t - s_{t,i} + s_b) \\ &\qquad + max_a Q(s_{t+1}, a)) \\ &= \alpha(d_t^i(s_t) + max_a Q(s_{t+1}, a)) \end{aligned} \tag{5}$$

Note that since this learning rule *is* Q-learning, albeit applied to a different reward structure, it shares all the convergens properties of Q-learning. In order to show that Equation 5 leads to good system level behavior, we need to show that agent $i$ maximizing $d_t^i(s_t)$ (e.g., following Equation5) will maximize the system reward $r_t$. Note that by definition $(s_t - s_{t,i} + s_b)$ is independent of the actions of agent $i$, since it is formed by moving agent $i$ to the absorbing state $s_b$ from which it cannot emerge. This effectively means the partial differential of $d_t^i(s_t)$ with respect to agent $i$ is[2]:

$$\begin{aligned} \frac{\partial}{\partial_i} d_t^i(s_t) &= \frac{\partial}{\partial_i} \left( r_t(s_t) - r_t(s_t - s_{t,i} + s_b) \right) \\ &= \frac{\partial}{\partial_i} r_t(s_t) - \frac{\partial}{\partial_i} r_t(s_t - s_{t,i} + s_b) \\ &= \frac{\partial}{\partial_i} r_t(s_t) - 0 \\ &= \frac{\partial}{\partial_i} r_t(s_t). \end{aligned} \tag{6}$$

Therefore any agent $i$ using a learning algorithm to maximize $d_t^i(s_t)$ will also maximize $r_t(s_t)$. Furthermore, note that QUICR-learning converges not only to a globally desirable solution (e.g., it statisfies the first property of being aligned with the system level goal), but it also converges faster since the rewards are more sensitive to the actions of agent $i$ because it removes much of the effects of the other agents through the counterfactual subtraction.

# 3. MULTI-AGENT GRID WORLD EXPERIMENTS

We performed a series of simulations to test the performance of Q-Learning, Local Q-Learning and QUICR-Learning for large-scale multi-agent systems. We selected the the multi-agent Grid World Problem, a variant of the standard Grid World Problem [7]. In this problem, at each time step, the agent can move up, down, right or left one grid square, and receives a reward (possibly zero) after each move. The observable state space for the agent is its grid coordinate and the reward it receives depends on the grid square to which it moves. In the episodic version, which is the focus of this paper, the agent moves for a fixed number of time steps, and then is returned to its starting location.

In this paper we compare learning algorithms in a multi-agent version of the grid world problem. In this instance of the problem there are multiple agents navigating the grid simultaneously influencing each others' rewards. In this problem agents are rewarded for observing tokens located in the grid. Each token has a value between zero and one, and each grid square can have at most one token. When an agent moves into a grid square it observes a token and receives a reward for the value of the token. Rewards are only received on the first observation of the token. Future observations from the agent or other agents do not receive rewards in the same episode. If two agents move into the same square at the same time. More precisely, $r_t$ is computed by summing the agents at the same location as unobserved tokens, weighted by the value of the tokens:

$$r_t(s_t) = \sum_i \sum_j V_j I^t_{s_{t,i}=L_j} \ . \tag{7}$$

where $I^t$ is the indicator function which returns one when an agent in state $s_{t,i}$ is in the location an unobserved token $L_j$. The global objective of the multi-agent Grid World Problem is to observe the highest aggregated value of tokens in a fixed number of time steps T.

## 3.1 Learning Algorithms

In each algorithm below, we use the TD(0) update rule. The standard Q-learning is based on the full reward $r_t$:

$$\Delta Q(s_t, a_t) = \alpha(r_t(s_t) + max_a Q(s_{t+1}, a)) \ . \tag{8}$$

[2]Though in this work we show this result for differentiable states, the principle applies to more general states, including discrete states.

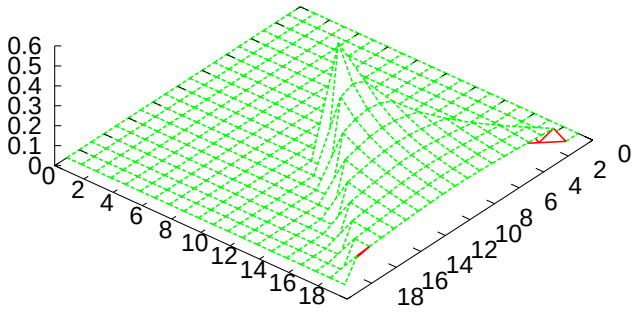


**Figure 1: Distribution of Token Values in "Cliff" World.**

Local Q-learning is only a function of the specific agent's own state:

$$\Delta Q_{loc}(s_t, a_t) = \alpha(\sum_j V_j I^{t,i}_{s_{t,i}=L_j} + max_a Q_{loc}(s_{t+1}, a)).$$

QUICR-learning instead updates with a reward that is a function of all of the states, but uses counterfactuals to suppress the effect of other agents' actions:

$$\Delta Q_{UICR}(s_t, a_t) = \alpha(r_t(s_t) - r_t(s_t - s_{t,i} + s_b) + max_a Q_{UICR}(s_{t+1}, a)),$$

where $s_t - s_{t,i} + s_b$ is the state resulting from removing agent $i's$ state and replacing it with the absorbing state $s_b$.

## 3.2 Results

To evaluate the effectiveness of QUICR-learning in the multi-agent Grid World, we conducted experiments on two different types of token distributions. The first set of tokens is designed to force congestion and tests the ability of QUICR-learning in domains where the reward function is far from being linearly separable. The second set is randomly generated from Gaussian kernels, to illustrate that the QUICR-learning capabilities in a non-hand crafted domain with spread out tokens (a domain favoring less dependend, local learners).

In all the experiments the learning rate was set to 0.5, the actions were chosen using an $\epsilon$-greedy ($\epsilon = 0.15$) exploration scheme and tables were initially set to zero with ties broken randomly.

## 3.3 Cliff World Token Value Distribution

The first experimental domain we investigated consisted of a world where the value of tokens was highest at the center and went down inversely with distance from the center. Three quarters of the surface was then set to zero, with only a one quarter "slice" remaining. Figure 1 conceptualizes this distribution for a 20x20 world.

Figure 2a shows the performance for 100 agents on a 400 unit-square world for the token value distribution shown in Figure 1, and where an episode consists of 20 time steps (error bars of $\pm$ one $\sigma$ are included, though in most cases they are smaller than the symbols). The performance measure in these figures is sum of full rewards ($r_t(s_t)$) received in an episode, normalized so that the maximum reward achievable is 1.0. Note all learning methods are evaluated on the same reward function, independent of the reward function that they are internally using to assign credit to the agents.

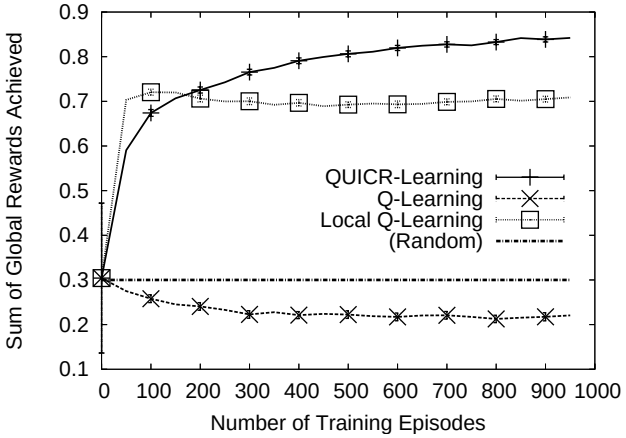


**Figure 2: Learning Rates in "Cliff" World with 100 Agents.**

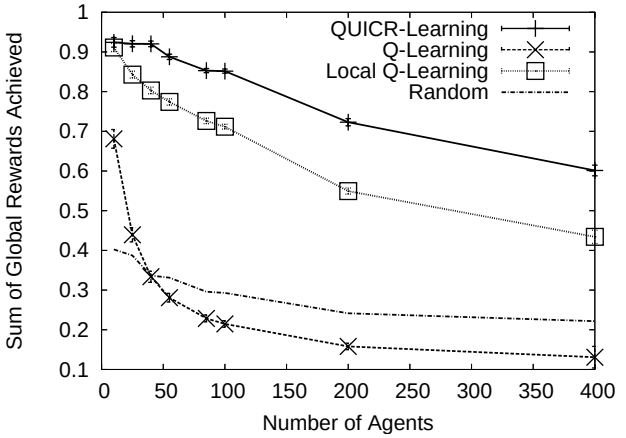


**Figure 3: Scaling Properties of Different Payoff Functions.**

The results show that local Q-learning generally produced poor results. This problem is caused by all agents aiming to acquire the most valuable tokens, and congregating towards the center of the world where such tokens are located. In essence, in this case agents using local Q-learning competed, rather than cooperated. The agents using standard Q-learning performed even worse, as the agents were plagued by the credit assignment problem associated with each agent receiving the full world reward for each individual action they took. In fact agents using local Q-learning performed even worse than agents taking random moves. This happens since these agents tend to form policies that are somewhat arbitrary, and do not even provide the amount of coverage achieved by a rover taking random actions. In contrast, agents using QUICR-learning learned rapidly, outperforming both local and standard Q-learning.

Figure 3 explores the scaling properties for each algorithm. As the number of agents was increased, the difficulty of the problem was kept constant by increasing the size of the grid-world, and allocating more time for an episode. Specifically the ratio of the number of agents to total number of grid squares and the ratio of the number of agents to total value of tokens was held constant. In addition the ratio of the fur-

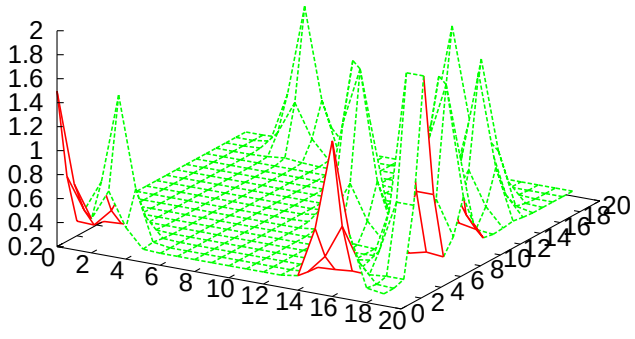

**Figure 4: Distribution of Token Values in "Random" World.**

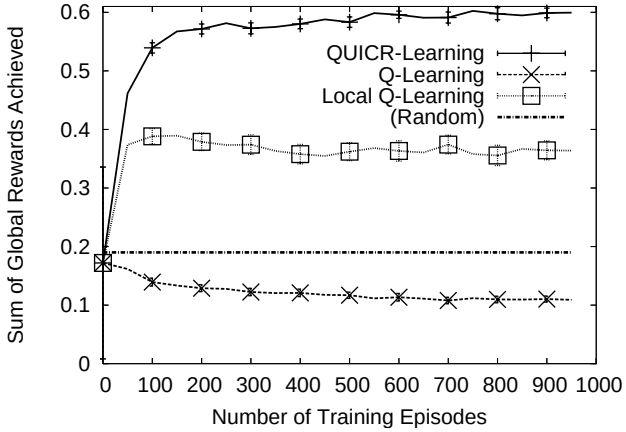


**Figure 5: Learning Rates in Random World with 100 Agents.**

thest grid square from the agents' starting point to the total amount of time in an episode was also held constant (e.g., 40 agents, 20x20 grid, 20 steps, 400 agents, 63x63 grid, 63 time steps). The scaling results show that agents using both local and standard Q-learning deteriorate rapidly as the number of agents increases. Agents using QUICR-learning on the other hand were not strongly affected by the increase in the size of the problem, and outperformed local and standard Q-learners. This is because QUICR-learning agents received rewards that were both aligned with the sytem goal had high agent sensitivity (i.e.,less affected by the size of the system). This result underscores the need for using rewards that suppress the affect of other agents actions in large systems.

## 3.4 Random World Token Value Distribution

In the second set of experiments, we investigate the behavior of agents in a gridworld where the token values are randomly distributed. In this world, for $n$ agents, there are $n/3$ Gaussian 'attractors' whose centers are randomly distributed. Figure 4 shows an instance of the gridworld using this distribution for the 20x20 world, used in the experiments with 100 agents.

The results in Figures 5 and 6 show that agents using QUICR-learning are insensitive to changes in the token value distribution. Agents using local Q-learning perform significantly better in this case, showing a much larger sensitivity to the token distribution. The improvements are due to the spreading of tokens over a larger area, which allows agents aiming (and failing) to collect high valued tokens to still collect mid to low-valued tokens, surrounding the high valued tokens. In this domain, agents using standard Q-learning performed particularly poorly. Indeed they were outperformed by agents performing random walks for systems of 100 agents. Due to the distributed tokens and the large number of agents, agents using standard Q-learning were never able to form an effective policy. These agents essentially moved from their initial random walk to a fixed arbitrary policy, causing them to spread out less than agents performing independent random walks, and thus perform slightly worse than random agents.

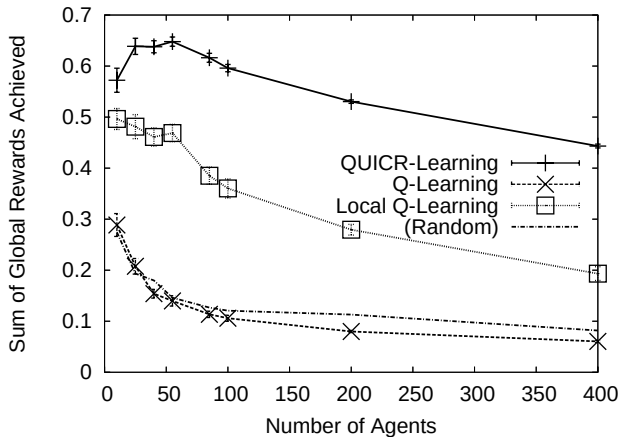


**Figure 6: Scaling Properties of Different Payoff Functions.**

The scaling results show again that QUICR-learning performs well as the number of agents increases. The interesting result here is that QUICR-learning does better with 40 to 55 agents than with 10 agents (this is not a statistical quirk, but a repeated phenomenon in many different random token configurations) . One potential cause is that the larger number of agents more efficiently explore the space without being beset by the problems encountered by the other learning algorithms. The scaling results confirm that standard Q-learners perform slightly coordinated random walks in this setting, performing ever so slightly worse than random in all cases with more than 25 agents. With this token distribution the rewards used in standard Q-learning appear to be no better than random rewards, even with ten agents. While local Q-learning performs better on this token distribution than the previous one, it still scales less gracefully that does QUICR-learning.

# 4. DISCUSSION

In large-scale multi-agent systems, using standard single-agent reinforcement learners is problematic because an agent will often have little influence over the reward it is trying to maximize. In our example problems, an agent's reward received after an action could be influenced by as many as 25,200 other actions from other time-steps and other agents. Even temporal difference methods that perform very well in single agent systems will be overwhelmed by the number of actions influencing a reward in the multi-

agent setting. To show that reinforcement learning can be used in large-scale problems, this paper introduced QUICR-learning, which aims at reducing the impact of other agent's actions without assuming linearly separable reward functions. Within the Q-learning framework, QUICR-learning uses the difference reward computed with immediate counterfactuals. While eliminating much of the influence of other agents, this reward was shown mathematically to be aligned with the global reward: agents maximizing the difference reward will also be maximizing the global reward. Experimental results in two Grid World problems with hundreds of agents, confirm the analysis showing that QUICR-learning learns in less time than standard Q-learning, and achieves better results than Q-learning variants that use local rewards and assume linear separability. While this method was used with TD(0) Q-learning updates, it also naturally extends to TD($\lambda$), Sarsa-learning and Monte Carlo estimation. In domains with difficult temporal credit assignment issues, the use of these other variants could be beneficial.

## 5. REFERENCES


[1] M. Dorigo and L. M. Gambardella. Ant colony systems: A cooperative learning approach to the travelling salesman problem. *IEEE Transactions on Evolutionary Computation*, 1(1):53–66, 1997.

[2] J. Hu and M. P. Wellman. Multiagent reinforcement learning: Theoretical framework and an algorithm. In *Proceedings of the Fifteenth International Conference on Machine Learning*, pages 242–250, June 1998.

[3] C. Jones and M. J. Mataric. Adaptive division of labor in large-scale multi-robot systems. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS-03)*, pages 1969–1974, Las Vegas, NV, July 2003.

[4] L. P. Kaelbling, M. L. Littman, and A. W. Moore. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 4:237–285, 1996.

[5] K. Konolige, D. Fox, C. Ortiz, A. Agno, M. Eriksen, B. Limketkai, B. Morisset J. Ko, D. Schulz, B. Stewart, and R. Vincent. Centibots: Very large scale distributed robotic teams. In *Proc. of the International Symposium on Experimental Robotics (ISER-04)*, Singapore, June 2004.

[6] P. Stone, R. S. Sutton, and G. Kuhlmann. Reinforcement learning for RoboCup-soccer keepaway. *Adaptive Behavior*, 2005.

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

[8] C. Watkins and P. Dayan. Q-learning. *Machine Learning*, 8(3/4):279–292, 1992.

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