Learning Sequences of Actions in Collectives of Autonomous Agents

Kagan Tumer
NASA Ames Research Center
Mailstop 269-3
Moffett Field, CA 94035
kagan@ptolemy.arc.nasa.gov

Adrian K. Agogino
The University of Texas
ENS 518
Austin, TX 78712
agogino@ece.utexas.edu

David H. Wolpert
NASA Ames Research Center
Mailstop 269-2
Moffett Field, CA 94035
dhw@ptolemy.arc.nasa.gov

ABSTRACT

In this paper we focus on the problem of designing a collective of autonomous agents that individually learn sequences of actions such that the resultant sequence of *joint* actions achieves a predetermined global objective. Directly applying Reinforcement Learning (RL) concepts to multi-agent systems often proves problematic, as agents may work at cross-purposes, or have difficulty in evaluating their contribution to achievement of the global objective, or both. Accordingly, the crucial design step in designing multi-agent systems focuses on how to set the rewards for the RL algorithm of each agent so that as the agents attempt to maximize those rewards, the system reaches a globally "desirable" solution. In this work we consider a version of this problem involving multiple autonomous agents in a grid world. We use concepts from collective intelligence [15, 23] to design rewards for the agents that are "aligned" with the global reward, and are "learnable" in that agents can readily see how their behavior affects their reward. We show that reinforcement learning agents using those rewards outperform both "natural" extensions of single agent algorithms and global reinforcement learning solutions based on "team games".

Categories and Subject Descriptors

I.2.11 [**Distributed Artificial Intelligence**]: Multiagent systems. I.2.6 Learning

General Terms

Design, Economics, Experimentation, Theory

Keywords

Reinforcement learning, MAS, Q-learning

1. INTRODUCTION

Many challenging problems involve coordinating a large number of autonomous agents to collectively address a well-defined, global, time-dependent task. Examples of such problems include controlling constellations of satellites, constructing distributed algorithms, routing over a data network, and controlling a collection of planetary exploration vehicles (e.g., rovers on Mars, or submersibles under Europa's ice caps). In each case, a single agent controlling everything would need too large of state space. For such problems there are two fundamental issues that need to be addressed:

- ensuring that the agents learn a *sequence* of actions that optimize each agent's "payoff utility function" (i.e., achieve a private goal); and
- ensuring that, as far as the provided "world utility function" is concerned, the agents do not work at cross-purposes (i.e., making sure that the private goals of the agents and the global goal are "aligned").

For single agent systems, the first of these issues has been dealt with extensively, and there are many learning systems, (e.g., Q-learners [17]) that have successfully been applied to real world problems [2]. The second problem has received less attention, and generally the solution consists of either each agent receiving the world utility as their payoff utility (e.g., "team" games [4]), or of imposing external mechanisms (e.g., contracts, auctions) that encourage the agents to work together [8, 11].

Addressing these two issues simultaneously is one of the main problems in designing multi-agent systems [9, 12]. If the agents are not designed to work well with each other, they may not learn their task properly, may interfere with each other's ability to contribute to the world utility, or simply perform useless repetitive work. Hand tailoring the agents' payoff functions may offer an alternative, but such systems: (i) have to be laboriously modeled; (ii) provide "brittle" global performance; (iii) are not "adaptive" to changing environments; and (iv) generally do not scale well.

To sidestep these problems, yet address the design requirements listed above (i.e., "alignedness" and "learnability") one can use the "COllective INtelligence" (COIN) framework [15, 21]. A COIN is a large multi-agent system where there is a well-defined "world utility" function which rates the possible dynamic histories of the collection and where there is little to no centralized control. We are particularly interested in the case where each agent is "selfish" and runs a Reinforcement Learning (RL) algorithm [14].

Given this framework, COIN theory addresses a new design problem: Assuming the individual agents are able to

AAMAS'02, July 15-19, 2002, Bologna, Italy.

maximize their own utility functions (e.g., through reinforcement learning), what set of payoff utilities for the individual agents will, when pursued by those agents, result in high world utility? In other words, how can we leverage an assumption that our learners are individually fairly good at what they do, to induce good collective behavior?

There are two quantifiable properties (discussed in detail in Section 2) that help answer this question. First, the utility functions for the individual agents need to be "aligned" with the world utility, in that an action taken by an agent that improves its payoff utility also improves the world utility. Second, the utility functions need to be "learnable" in that an agent has to be able to discern the effect of its actions on its utility and select actions that optimize that utility. As we will highlight below, COIN theory provides utilities for individual agents that maximize the second property while satisfying the first one.

A canonical example of a naturally occurring system that can be viewed as a COIN is a human economy. One can take the agents to be the individuals trying to maximize their payoff utilities (e.g., maximize bank account, advance career). One might then take the time average of the gross domestic product as the world utility ("world utility" is not a construction internal to a human economy, but rather something defined from the outside). To achieve high world utility it is necessary to avoid having the agents work at cross-purposes lest frustrational phenomena like the tragedy of the commons occur, in which individual avarice works to lower world utility [7]. One way to avoid such phenomena is by modifying the agents' utility functions via punitive legislation, in essence making sure the agents' utility functions are aligned with the world utility. Securities and Exchange Commission (SEC) regulations designed to prevent insider trading can be viewed as a real world example of an attempt to make such a modification to the agents' utilities. For example, a trade that once may have added to your wealth while hurting the economy, may now lead to your prosecution. You are therefore unlikely to make such a trade. Your utility and the world utility have become more aligned.

In designing a COIN we have more freedom than the SEC though, in that there is no base-line "organic" payoff utility function over which we must superimpose legislation-like incentives. Rather, the entire "psychology" of the individual agents is at our disposal when designing a COIN. This freedom is a major strength of the COIN approach, in that it obviates the need for honesty-elicitation mechanisms, like auctions, which form a central component of conventional economics.

The COIN design problem is related to work in many fields beyond multiagent systems and computational economics, including mechanism design, reinforcement learning for adaptive control, computational ecologies, and game theory. However none of these fields directly addresses the inverse problem of how to design the agents' utilities to reach a desirable world utility value in its full generality. This is even true for the field of mechanism design, which while addressing an inverse problem similar to that of COIN design, does so only for certain restricted domains, and does not address the "learnability" issue. (Mechanism design is mostly appropriate when there are pre-specified goals underlying agents' utilities over which "incentives" need to be provided, and when Pareto-optimality (rather than optimization of a world utility) is often the goal [21].)

The COIN framework has been successfully applied to multiple domains including packet routing over a data network [20] and the congestion game known as Arthur's El Farol Bar problem [23]. In particular, in the routing domain, the COIN approach achieved performance improvements of a factor of three over the conventional Shortest Path Algorithm (SPA) routing algorithms currently running on the internet [19], and avoided the Braess' routing paradox which plagues the SPA-based systems [15].

In the work described above, agents were concerned with optimizing "rewards" (i.e., utility value of a single time step). In this paper we extend these results to a problem where agents need to optimize a time-extended utility function through selecting sequences of actions. We show that in this significantly more complex domain, agents that use COIN theory-based utilities provided solutions that are significantly superior to agents that either use team games or "natural" utilities. In Section 2, we provide some background on COIN-theoretic concepts and highlight relevant theoretical developments. In Section 3, we describe the problem domain and develop the COIN solution to this problem. In Section 4, we present and discuss the simulation results. Finally in Section 5, we provide a simple example that demonstrates how and why COIN-theory based algorithms significantly outperformed more "natural" or "traditional" approaches.

2. BACKGROUND: COLLECTIVE INTELLIGENCE

In this section, we summarize the portion of COIN theory needed to describe the learning of sequences of actions in a distributed system [21]. Let Z be an arbitrary vector space whose elements ζ give the joint move of all agents in the system (i.e., ζ specifies the full "worldline" consisting of the actions/states of all the agents). The provided **world utility** $G(\zeta)$, is a function of the full worldline, and we wish to search for the ζ that maximizes $G(\zeta)$.

In addition to G, for each agent η, there is a **payoff utility functions** $\{g_\eta\}$. The agents will act to improve their individual payoff functions, even though, we, as system designers are only concerned with the value of the world utility G. To specify all agents other than η, we will use the notation $\hat{}\eta$.

2.1 Intelligence

We need to have a way to "standardize" utility functions so that the numeric value they assign to a ζ only reflects the ranking of ζ relative to certain other elements of Z. We call such a standardization of some arbitrary utility U for agent η the "**intelligence** for η at ζ with respect to U". Here we use intelligences that are equivalent to percentiles:

$$\epsilon_U(\zeta : \eta) \equiv \int d\mu_{\zeta_{\hat{}\eta}}(\zeta')\Theta[U(\zeta) - U(\zeta')] \,, \qquad (1)$$

where the Heaviside function Θ is defined to equal 1 when its argument is greater than or equal to 0, and to equal 0 otherwise, and where the subscript on the (normalized) measure $d\mu$ indicates it is restricted to ζ' sharing the same non-η components as ζ.[1] Note that intelligence value are

[1]The measure must reflect the type of system at hand, e.g., whether Z is countable or not, and if not, what coordinate system is being used. Other than that, any convenient choice

always between 0 and 1. Intuitively, intelligence values indicate what percentage of η's actions would have resulted in lower utility. Accordingly, $\epsilon_{g_\eta}(\zeta : \eta) = 1$ means that agent η is fully rational at ζ, in that its move maximizes its payoff, given the moves of other agents. Figure 1 shows an example where 60% of η's actions would have resulted in worse utility, giving η an intelligence of 0.6 at that point (ζ).

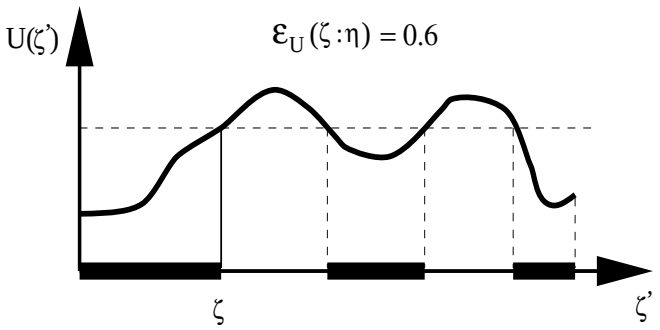


Figure 1: Intelligence of agent η at state ζ for utility U: ζ is the actual joint move at hand. The x-axis shows agent η's alternative possible moves (all states ζ' having ζ's values for the moves of all agents other than η.). The bold lines show the alternative moves that η could have made that would have given η a worse value of the utility U. The fraction of those bold lines to the full set of η's possible moves (which is 0.6 in this example) is the intelligence of agent η at ζ for utility U, denoted by $\epsilon_U(\zeta : \eta)$.

Our uncertainty concerning the behavior of the system is reflected in a probability distribution over Z. Our ability to control the system consists of setting the value of some characteristic of the collection of agents, e.g., setting the payoff functions of the agents. Indicating that value by s, our analysis revolves around the following central equation for $P(G \mid s)$, which follows from Bayes' theorem:

$$P(G \mid s) = \int d\vec{\epsilon}_G P(G \mid \vec{\epsilon}_G, s) \int d\vec{\epsilon}_g P(\vec{\epsilon}_G \mid \vec{\epsilon}_g, s) P(\vec{\epsilon}_g \mid s) \ , \quad (2)$$

where $\vec{\epsilon}_g \equiv (\epsilon_{g_{\eta_1}}(\zeta : \eta_1), \epsilon_{g_{\eta_2}}(\zeta : \eta_2), \cdots)$ is the vector of the intelligences of the agents with respect to their associated payoff functions, and $\vec{\epsilon}_G \equiv (\epsilon_G(\zeta : \eta_1), \epsilon_G(\zeta : \eta_2), \cdots)$ is the vector of the intelligences of the agents with respect to G.

Note that, from a game-theoretic perspective, a point ζ where all players are rational, ($\epsilon_{g_\eta}(\zeta : \eta) = 1$ for all agents η, is a game theory Nash equilibrium [21]. On the other hand, a ζ at which all components of $\vec{\epsilon}_G = 1$ is a local maximum of G (or more precisely, a critical point of the $G(\zeta)$ surface).

If we can choose s so that the third conditional probability in the integrand, $P(\vec{\epsilon}_g \mid s)$, is peaked around vectors $\vec{\epsilon}_g$ all of whose components are close to 1 (that is agents are able to "learn" their tasks), then we have likely induced large payoff utility intelligences. If we can also have the second term, $P(\vec{\epsilon}_G \mid \vec{\epsilon}_g, s)$, be peaked about $\vec{\epsilon}_G$ equal to $\vec{\epsilon}_g$ (that is the payoff and world utilities are aligned), then $\vec{\epsilon}_G$ will also be large. Finally, if the first term in the integrand, $P(G \mid \vec{\epsilon}_G, s)$, is peaked about high G when $\vec{\epsilon}_G$ is large, then our choice of s will likely result in high G, as desired.

of measure may be used and the theorems will still hold.

2.2 Factoredness and Learnability

The requirement that payoff functions have high "signal-to-noise" (an issue not considered in conventional work in mechanism design) arises in the third term. It is in the second term that the requirement that the payoff functions be "aligned with G" arises. In this work we concentrate on these two terms, and show how to simultaneously set them to have the desired form.

Details of the stochastic environment in which the collection of agents operate, together with details of the learning algorithms of the agents, are reflected in the distribution $P(\zeta)$ which underlies the distributions appearing in Equation 2. Note though that *independent of these considerations*, our desired form for the second term in Equation 2 is assured if we have chosen payoff utilities such that $\vec{\epsilon}_g$ equals $\vec{\epsilon}_G$ exactly *for all* ζ. We call such a system **factored**. In game theory language, the Nash equilibria of a factored system are local maxima of G. In addition to this desirable equilibrium behavior, factored systems also automatically provide appropriate off-equilibrium incentives to the agents (an issue rarely considered in the game theory / mechanism design literature).

As a trivial example, any "team game" in which all the payoff functions equal G is factored [4]. However team games often have very poor forms for term 3 in Equation 2, forms which get progressively worse as the size of the system grows. This is because for large systems where G sensitively depends on all components of the system, each agent may experience difficulty discerning the effects of its actions on G. As a consequence, each η may have difficulty achieving high g_η in a team game. We can quantify this signal/noise effect by comparing the ramifications on $g_\eta(\zeta)$ arising from changes to ζ_η with the ramifications arising from changes to $\zeta_{\neg\eta}$ (i.e., changes to all nodes *other* than η). We call this quantification the differential **learnability** of payoff utility g_η, in the vicinity of ζ [22]:

$$\lambda_{\eta, g_\eta}(\zeta) \equiv \frac{\|\vec{\nabla}_{\zeta_\eta} g_\eta(\zeta)\|}{\|\vec{\nabla}_{\zeta_{\neg\eta}} g_\eta(\zeta)\|} \ . \quad (3)$$

The denominator in Equation 3 reflects how sensitive $g_\eta(\zeta)$ is to changes in $\zeta_{\neg\eta}$, or changes to agents other than η. In contrast, the numerator reflects how sensitive $g_\eta(\zeta)$ is to changing ζ_η. So at a given state ζ, the higher the learnability, the more $g_\eta(\zeta)$ depends on the move of agent η, i.e., the better the associated signal-to-noise ratio for η. Intuitively then, higher learnability means it is easier for η to achieve a large value of its intelligence.

2.3 Difference Utilities

Consider **difference** utilities, which are of the form:

$$U(\zeta) = G(\zeta) - \Gamma(f(\zeta)) \ , \quad (4)$$

where $\Gamma(f)$ is independent of ζ_η. Such difference utilities are factored [21]. In addition, under usually benign approximations, the differential learnability can be maximized over the set of difference utilities by choosing $f \equiv G$ and setting Γ to the expected value operator [21]. We call the resultant difference utility the **Aristocrat** Utility (AU):

$$AU(\zeta) = G(\zeta) - E(G \mid \zeta_{\neg\eta}, s) \ . \quad (5)$$

If possible, we would like each agent η to use the associated AU as its payoff function to ensure good form for both

$$
\begin{array}{c} \\ \eta_1 \\ \eta_2 \\ \eta_3 \\ \eta_4 \end{array}
\begin{array}{c} \zeta \\ \left[\begin{array}{ccc} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{array}\right] \end{array}
\quad \Longrightarrow \begin{array}{c} \text{Clamp } \eta_2 \\ \text{to "null"} \end{array} \quad
\begin{array}{c} (\zeta_{\hat{\eta}_2}, \vec{0}) \\ \left[\begin{array}{ccc} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{array}\right] \end{array}
$$

Figure 2: This example shows the impact of the clamping operation on the joint state of a four-agent system where each agent has three possible actions, and each such action is represented by a three-dimensional unary vector. The first matrix represents the joint state of the system ζ where agent 1 has selected action 1, agent 2 has selected action 3, agent 3 has selected action 1 and agent 4 has selected action 2. The second matrix displays the effect of clamping agent 2's action to the "null" vector (i.e., replacing ζ_{η_2} with $\vec{0}$).

terms 2 and 3 in Equation 2. This is not always feasible however. The problem is that to evaluate the expectation value defining its AU each agent needs to evaluate the current probabilities of each of its potential moves. However if the agent then changes its payoff function to be the associated AU it will in general substantially change its ensuing behavior. (The agent now wants to choose moves that maximize a different function from the one it was maximizing before.) In other words, it will change the probabilities of its moves, which means that its new payoff function is in fact not the AU for its actual (new) probabilities.

There are ways around this self-consistency problem, but in practice it is often easier to bypass the entire issue, by giving each η a payoff function that does not depend on the probabilities of η's own moves. One such payoff function is the **Wonderful Life** Utility (WLU). The WLU for agent η is parameterized by a pre-fixed **clamping parameter** CL_η chosen from among η's legal or illegal moves:

$$WLU_\eta \equiv G(\zeta) - G(\zeta_{\hat{\eta}}, CL_\eta) \; . \tag{6}$$

WLU is factored no matter what the choice of clamping parameter. Furthermore, while not matching the high learnability of AU, WLU usually has far better learnability than does a team game.

Figure 2 provides an example of clamping. As in that example, in many circumstances there is a particular choice of clamping parameter for agent η that is a "null" move for that agent, equivalent to removing that agent from the system, hence the name of this payoff function. For such a clamping parameter WLU is closely related to the economics technique of "endogenizing a player's (agent's) externalities" [10]. Indeed, WLU has conceptual similarities to Vickrey tolls [16] in economics, and Groves' mechanism [6] in mechanism design. However, because WLU can be applied to arbitrary, time-extended utility functions, and need not be restricted to the "null" clamping operator interpretable in terms of "externality payments", it can be viewed a generalization of these concepts.

It can be proven that in many circumstances, especially in large problems, $\lambda_{\eta,WLU}(\zeta) \geq \lambda_{\eta,G}(\zeta)$, i.e., WLU has higher differential learnability than does the team game choice of payoff utilities [21]. This is mainly due to the second term of WLU which removes a lot of the effect of other agents (i.e., noise) from η's utility. The result is that convergence to optimal G with WLU is much quicker (up to orders of magnitude so [21]) than with a team game. Furthermore, it is possible to use a WLU with a clamping parameter that is as close as possible to the expected action (and not the "null" action), which is roughly akin to a mean-field approximation to AU [21]. Such a WLU often provides higher learnability than does clamping to the null move [21].

Intuitively, one can look at AU and WLU from the perspective of a human company, with G the "bottom line" of the company, the agents η identified with the employees of that company, and the associated g_η given by the employees' performance-based compensation packages. For example, for a "factored company", each employee's compensation package contains incentives designed such that the better the bottom line of the corporation, the greater the employee's compensation. As an example, the CEO of a company wishing to have the payoff utilities of the employees be factored with G may give stock options to the employees. The net effect of this action is to ensure that what is good for the employee is also good for the company. In addition, if the compensation packages have "high learnability", the employees will have a relatively easy time discerning the relationship between their behavior and their compensation. In such a case the employees will both have the incentive to help the company and be able to determine how best to do so. Note that in practice, providing stock options is usually more effective in small companies than in large ones. This makes perfect sense in terms of the COIN formalism, since such options generally have higher learnability in small companies than they do in large companies, in which each employee has a hard time seeing how his/her moves affect the company's stock price.

3. MULTI-AGENT GRID WORLD PROBLEM

3.1 Problem Description

A common reinforcement learning problem is the Grid World Problem [14], where an agent navigates about a two-dimensional $n \times n$ grid. At each time step, the agent can move up, down, right or left one grid square, and receives a reward 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. This problem typically requires the use of a reinforcement learner that can optimize a sum of rewards in contrast to one that optimizes an immediate reward, since the agent may have to cross squares of low reward value to enter the squares of high value. Q-learners or the Sarsa algorithm [14] are often used for this problem. In this paper we apply COIN theory to a multi-agent version of the Grid World Problem. In this problem there are multiple agents navigating the grid simultaneously interacting with each others' rewards. This reward interaction is modeled through the use of tokens that are distributed throughout the grid squares of the grid world (Figure 3). 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 receives a reward for the value of the token and then removes the token so that a reward will no longer be received when an agent enters the grid square. However, all

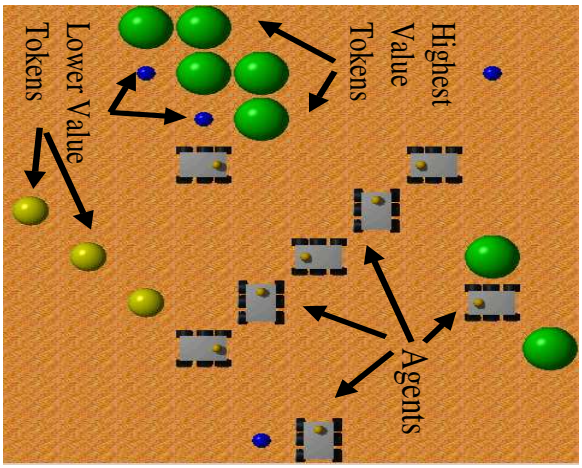


Figure 3: Agents collecting tokens of varying value.

the tokens are reset at the end of an episode. The global objective of the Multi-agent Grid World Problem is to collect the highest aggregated value of tokens in a fixed number of time steps.

The Multi-agent Grid World Problem is an idealized version of many real world problems, including the control of multiple planetary exploration vehicles (e.g., rovers on the surface of Mars, collecting rocks in an attempt to maximize total scientific return, submersible under Europa examining potential life signs). Furthermore, the agent interaction provides a critical study of coordination and interference, as the agents have the potential to work at cross-purposes. This problem can also exhibit the tragedy of the commons [7], where each agent attempting to maximize its own utility can drive the world utility to severely sub-optimal values. As such, the design of the payoff functions is crucial in this problem, and we address this issue below.

3.2 COIN Solution

To pose the Multi-agent Grid World Problem in the form of the COIN framework we need to define:

- $L_{\eta,t}$: The matrix representing the location of an agent. If agent η at time t is in location (x, y), then $L_{\eta,t,x,y} = 1$; otherwise $L_{\eta,t,x,y} = 0$. Furthermore, $\{L_{\eta,t}\}$ denotes the set all the agents' location matrices.

- $L^a_{\eta,t}$: The location matrix agent η would have had at time t, had it taken action a at time step $t-1$.

- L_η: The location matrix of agent η across all time ($L_\eta = \sum_t L_{\eta,t}$).

- $L_{\eta,<t}$: The location matrix of agent η across times less than t ($L_{\eta,<t} = \sum_{t'<t} L_{\eta,t'}$).

- L: The location matrix of all agents across all time ($L = \sum_\eta L_\eta = \sum_t \sum_\eta L_{\eta,t}$).

- $L_{<t}$: The location matrix of all agents across times less than t ($L_{<t} = \sum_{t'<t} L'_t = \sum_{t'<t} \sum_\eta L_{\eta,t'}$).

- $L_{\hat{\eta}}$: The location matrix of all agents other than η across all time ($L_{\hat{\eta}} = L - L_\eta$).

- $L_{\hat{\eta},<t}$: The location matrix of all agents other than η across times less than t ($L_{\hat{\eta},<t} = L_{<t} - L_{\eta,<t}$).

- Θ: The initial value and location of all tokens.

The space $\mathbf{Z}$ is composed of Θ and the set of all possible location matrices, $\{L_{\eta,t}\}$, given the length of an episode. A worldline ζ is a point in this space, i.e., the combination of the token configurations Θ, along with a particular set $\{L_{\eta,t}\}$. We now define the function $V(L, \Theta)$ which returns the value of a token received from a location matrix. Formally:

$$V(L, \Theta) = \sum_{x,y} \Theta_{x,y} \min(1, L_{x,y}). \tag{7}$$

The global utility $G(\zeta)$ is the sum off all the tokens collected during an episode:

$$G(\zeta) = V(L, \Theta). \tag{8}$$

Based on the definitions and world utility given above, let us now derive the COIN-based utility functions for this domain. In this formulation, the AU (given in Equation 5) become:

$$AU_\eta(\zeta) = G(\zeta) - \sum_{\vec{a} \in \vec{A}_\eta} p_{\vec{a}} V(L_\eta + L^{\vec{a}}_\eta, \Theta) \tag{9}$$

where A_η is the set of possible action sequences agent η can take. The second term in the equation is the expected value of the global utility over all the possible actions of agent η.

Now, let us formulate the WL utilities for this domain. First, setting the clamping parameter CL_η to the null vector, we obtain WL utility where the agent is removed from the worldline:

$$WLU^{\vec{0}}_\eta(\zeta) = G(\zeta) - V(L_{\hat{\eta}}, \Theta). \tag{10}$$

This utility returns an agent's contribution to the world utility. Note, this utility differs from one where the values of the tokens present in the locations visited by the agent are summed (i.e., a selfish utility). $WLU^{\vec{0}}$ gives the value of the tokens *in locations not visited by other agents*, i.e., the values of token that would not have been picked up had agent η not been in the system.

Because these utilities are based on the performance on a full episode, they are problematic to work with directly. We therefore introduce single time step "rewards" that will help in learning the set of actions (e.g., through Q-learners or Sarsa learners) that will lead to good values for the utility. Note, that the utilities will be undiscounted sums of these rewards. To that end, first let us decompose an arbitrary utility U in the following manner:

$$U(L) = \sum_t U(L_{<t+1}) - U(L_{<t}). \tag{11}$$

A single time step reward R_t is simply:

$$R_t(L) = U(L_{<t+1}) - U(L_{<t}) \tag{12}$$

Now we can generate the three single time step reward

versions of the three utilities[2]:

$$GR_t(\zeta) = V(L_{<t+1}, \Theta) - V(L_{<t}, \Theta) \quad (13)$$

$$AR_{\eta,t}(\zeta) = GR_t(\zeta) - \sum_{\vec{a} \in \vec{A}_\eta} p_{\vec{a}}(V(L_{\eta,<t+1} + L^{\vec{a}}_{\eta,<t+1}, \Theta) - V(L_{\eta,<t} + L^{\vec{a}}_{\eta,<t}, \Theta)) \quad (14)$$

$$WLR^{\vec{0}}_{\eta,t}(\zeta) = GR_t(\zeta) - (V(L_{\eta,<t+1}, \Theta) - V(L_{\eta,<t}, \Theta)) \quad (15)$$

Unfortunately the formulation for AR has the drawback that the set of all possible action sequences is very large, and grows exponentially with t. This issue is resolved in this paper by taking an approximation that uses only the action in the last time step instead of the entire sequence of actions.

4. RESULTS

To evaluate the effectiveness of the COIN approach in the Multi-agent Grid World, we conducted experiments where the agents used four different utility functions. The first of these was the Selfish Utility (SU), where each agent receives the weighted total of the tokens that it alone collected. It is the natural extension of the single agent problem, and represents the optimal utility in the single rover domain. The second utility was the Team Game (TG) utility where each agent received the full world utility. The third utility was the WLU, which represents the contribution an agent made to the token collection, by looking at the difference in the total token collection with and without that agent. The fourth and final utility was AU, where the agent's contribution is computed as the difference between the action it took and its expected action.

Each agent was controlled by a Q-learner. A learner's input space consisted of the location of the agent in the grid world, and the action space was the four directions that the agent could move. The Q-tables were initially set to zero, and the discounting parameter was set to 0.95. Actions were chosen stochastically based on Q-values, where the probability that an agent took action a_i given state $s = \frac{k^{Q(s,a_i)}}{\sum_j k^{Q(s,a_j)}}$. In our experiments k was set to 50.

Figure 4 shows the results for 10 agents on a 100 unit-square grid where an episode consists of 10 time steps (including error bars of $\pm$ one σ). The results showed that SU produced poor results, results that were indeed worse than random actions. This is caused by all agents aiming to acquire the most valuable tokens, in effect competing rather than cooperating. The agents using TG fared better, but their learning was slow. This system was plagued by the signal-to-noise problem associated with each agent receiving the full world reward for each individual action they took. Notice both the selfish agents and those trained with TG had a drop in their performance in the early going, as they learned the "wrong" actions. Team game agents overcame this early setback whereas selfish agents never did. In contrast, agents using WLU and AU performed almost optimally, because the reinforcement signal they received more clearly showed how their actions affected the world reward.

[2]In the actual implementation there are some tie breaking rules if more than one agent goes into the same square at the same time.

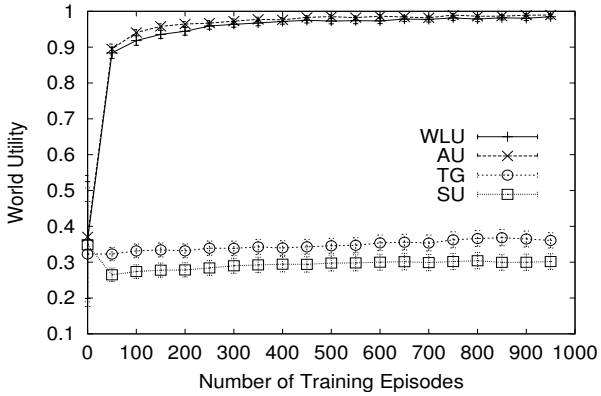


Figure 4: Effect of Payoff Utility on System Performance (10 agents on a 10x10 grid).

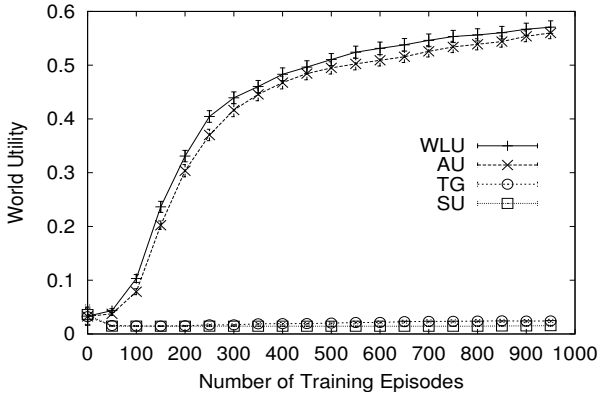


Figure 5: Effect of Payoff Utility on System Performance (100 rovers on a 32x32 grid).

Figure 5 shows results for 100 agents on a 1024 unit-square grid where an episode consists of 32 time steps. Qualitatively, the results are similar to the 10 agent case. However, note that the team game agents have a harder time learning, because in this case the reinforcement signal is even further diluted. We explore this scaling issue in more detail in Figure 6. With very few agents, the selfish learners did not compete with each other as much and were able to obtain acceptable results. Their performance however, deteriorated rapidly, when the number of agents in the system increased. Similarly, agents using the team game reward were not hampered as much by the noise associated with other agents when the number of agents was low. As the system scaled up however, only the WLR-trained agents were able to operate collectively. This underscores the need for a utility that has good signal-to-noise properties so that the agents have an opportunity to learn the actions that will optimize their utilities.

5. NASH EQUILIBRIA AND WORLD UTILITY OPTIMA

In this simple example, we demonstrate how the Nash equilibrium of the system where each agent uses the WLU coincides with the world utility optimum, and how agents optimizing their Wonderful Life Utility will optimize the world utility (Figure 7). Suppose that two agents are on

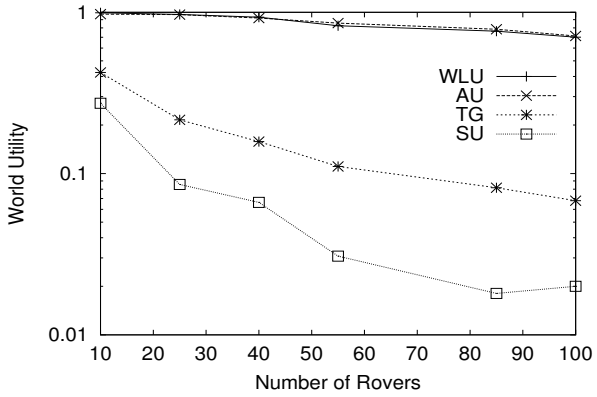
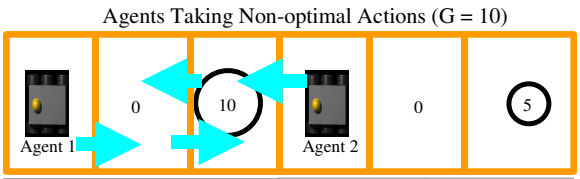


Figure 6: Scaling Properties of Different Payoff Functions.

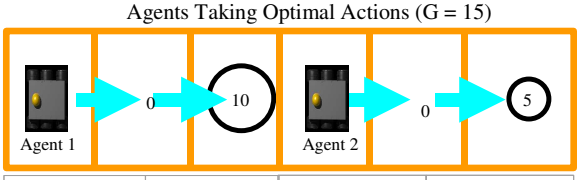
a six square world, can move left and right and can take actions for two time steps. There are two tokens, one of values 5 and the other of value 10 that the agents can pick up by entering the appropriate square. A plausible, yet non-optimal set of actions consists of agent 1 moving right twice and agent 2 moving left first and then taking an arbitrary action. Note that this is a Nash equilibrium for the system where the agents use the selfish reward described in Section 3. In this scenario agent 2 will pick up a token worth 10 on its first time step and no tokens on the second time step. Agent 1 will not pick up any tokens. This results in a world reward of 10 for the first time step and 0 for the second, resulting in a world utility of 10, which is not optimal.

Now, let us look at agent 2's WL payoff for this set of moves: For the first time step, the WL reward turns out to be the same the the selfish reward: agent 2 receives 10 for picking up the token. The WLR for agent 2 in the second time step is more interesting. The first parameter of the V function, L_η, now does not include agent 2, causing this function do disregard any tokens agent 2 previously picked up. This causes the V function to report that the token of value 10 is still available in the second time step. Since agent 1 moves into the square with this token in the second time step, it receives credit for picking up the token, meaning the world reward without agent 2 is 10 for this time step. Because the world reward with agent 2 was 0 (token picked up previous time step), the WL reward for agent 2 for the second time step is $WLR_{\eta_2,t=2} = 0 - 10 = -10$. Intuitively the WLR can be thought of as an agent's contribution to the world reward at this time step. Since at $t = 1$ agent 2 picked up a token that could have been picked up at $t = 2$, it had a deleterious effect on the second time step.

Now the time-extended WLU for agent 2 can be computed by summing the WLRs. This results in a WLU of 0, even though agent 2 picks up a token weighted 10 (10 for t=1 and -10 for t=2). The interpretation for this "counter-intuitive" utility value is clear: because that token would have been picked up by agent 1 at another time step, the net effect of agent 2's actions on the world utility was nil, resulting in a WLU value of 0. Because moving to the right twice provides a WLU value of 5 for agent 2, an agent optimizing its WL payoff utility will take this second action. Similarly agent 1 moving right twice will receive a WLU of 10. As this simple example shows, each agent maximizing its WLU leads the system to the world utility maximum where both tokens are



	G	WLU η_1	WLU η_2	SU η_1	SU η_2
t=1	10	0	10	0	10
t=2	0	0	**-10**	0	0
Total	10	0	**0**	0	10



	G	WLU η_1	WLU η_2	SU η_1	SU η_2
t=1	0	0	0	0	0
t=2	15	10	5	10	5
Total	15	10	**5**	10	5

Figure 7: WLU Nash Equilibria and World Utility Optima

picked up.

Let us analyze the game-theoretic "equilibrium" solution for WLU and SU in these two solutions: The SU is in a Nash equilibrium for the first set of moves, in that neither agent can improve its SU by unilaterally changing its actions. Therefore, the system is "stuck" in this suboptimal solution. Furthermore, even if the agents stumble upon the second solution by accident, they will not remain there, as this solution is unstable with payoff utilities given by SU: Agent 2 can change its move (in future episodes) and improve its payoff utility from 5 to 10. That this move reduces agent 1's utility from 10 to 0, and the world utility from 15 to 10 has no influence on agent 2's actions. Note, however, that agents with WLU as their payoff utilities are in an equilibrium state in the second set of actions. They will therefore seek this solution as it offers higher payoff utilities for each agent. The use of WLU has the net effect of "aligning" the Nash equilibrium of the agents with the world utility optimum, ensuring that when the agents optimize their payoff utilities, the world utility is also at a local – and in this case also the global – optimum.

6. DISCUSSION

In this work we focus on the problem of designing a collective of autonomous agents that individually learn sequences of actions such that the resultant sequence of joint actions achieves a predetermined global objective. In particular we discuss the problem of controlling multiple agents in a grid world, a problem related to many real world problems including exploration vehicles trying to maximize aggregate scientific data collection (e.g., rovers on the surface of Mars). In this domain, we addressed the critical issue of what utility functions those agents should strive to maximize. We

extended previous results on collective intelligence to agents attempting to maximize sequences of actions, and used Q-learning with rewards set by COIN theory. Our results demonstrate that RL rovers using COIN-derived goals outperform both "natural" extensions of single agent algorithms and global reinforcement learning solutions based on "team games".

Our investigations revealed an interesting situation where the theoretically "best" strategy was not necessarily the best approach in practice. Although AU is theoretically superior to WLU (higher learnability), two issues prevent us from fully exploiting its power: First, the "expected" action is impossible to compute in a time extended setting, since even a simple case where an agent has four actions and ten time steps leads to 4^{10} possible actions. Even Monte Carlo sampling of such a space will yield highly inaccurate estimates of the potential actions and their rewards. Second, estimating the correct probability distributions over the possible actions causes the utility values to change, creating a self-consistency problem. To sidestep both issues, in this article we chose to focus on the last time step (e.g., current step for the agent) and approximated the AU with the agent taking each of the four actions possible in that time step with equal likelihood. The resulting utility function provided good solutions, but the performance of such a "handicapped" AU did not exceed those of the conceptually simpler WLU.

Future work in this area includes investigating efficient AU computation for sequences of actions, and investigating the "mean field" approximation to AU by clamping the actions of an agent to the average action discussed in Figure 2. This approach avoids both difficulties associated with the proper AU, and has been shown to lead to good world utility values in single step reward maximization problems [21].

7. REFERENCES

[1] C. Boutilier. Multiagent systems: Challenges and opportunities for decision theoretic planning. *AI Magazine*, 20:35–43, winter 1999.

[2] J. A. Boyan and M. Littman. Packet routing in dynamically changing networks: A reinforcement learning approach. In *Advances in Neural Information Processing Systems - 6*, pages 671–678. Morgan Kaufman, 1994.

[3] C. Claus and C. Boutilier. The dynamics of reinforcement learning cooperative multiagent systems. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence*, pages 746–752, Madison, WI, June 1998.

[4] R. H. Crites and A. G. Barto. Improving elevator performance using reinforcement learning. In D. S. Touretzky, M. C. Mozer, and M. E. Hasselmo, editors, *Advances in Neural Information Processing Systems - 8*, pages 1017–1023. MIT Press, 1996.

[5] A. Greenwald, E. Friedman, and S. Shenker. Learning in network contexts: Experimental results from simulations. *Journal of Games and Economic Behavior: Special Issue on Economics and Artificial Intelligence*, 35(1/2):80–123, 2001.

[6] T. Groves. Incentives in teams. *Econometrica*, 41:617–631, 1973.

[7] G. Hardin. The tragedy of the commons. *Science*, 162:1243–1248, 1968.

[8] 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.

[9] N. R. Jennings, K. Sycara, and M. Wooldridge. A roadmap of agent research and development. *Autonomous Agents and Multi-Agent Systems*, 1:7–38, 1998.

[10] W. Nicholson. *Microeconomic Theory*. The Dryden Press, seventh edition, 1998.

[11] T. Sandholm and R. Crites. Multiagent reinforcement learning in the iterated prisoner's dilemma. *Biosystems*, 37:147–166, 1995.

[12] S. Sen. *Multi-Agent Learning: Papers from the 1997 AAAI Workshop (Technical Report WS-97-03*. AAAI Press, Menlo Park, CA, 1997.

[13] P. Stone and M. Veloso. Multiagent systems: A survey from a machine learning perspective. *Autonomous Robots*, 8(3), 2000.

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

[15] K. Tumer and D. H. Wolpert. Collective intelligence and Braess' paradox. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence*, pages 104–109, Austin, TX, 2000.

[16] W. Vickrey. Counterspeculation, auctions and competitive sealed tenders. *Journal of Finance*, 16:8–37, 1961.

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

[18] M. P. Wellman. A market-oriented programming environment and its application to distributed multicommodity flow problems. In *Journal of Artificial Intelligence Research*, 1993.

[19] D. H. Wolpert, S. Kirshner, C. J. Merz, and K. Tumer. Adaptivity in agent-based routing for data networks. In *Proceedings of the fourth International Conference of Autonomous Agents*, pages 396–403, 2000.

[20] D. H. Wolpert and K. Tumer. An Introduction to Collective Intelligence. Technical Report NASA-ARC-IC-99-63, NASA Ames Research Center, 1999. URL:`http://ic.arc.nasa.gov/ic/projects/coin_pubs.html`. To appear in Handbook of Agent Technology, Ed. J. M. Bradshaw, AAAI/MIT Press.

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

[22] D. H. Wolpert, K. Tumer, and J. Frank. Using collective intelligence to route internet traffic. In *Advances in Neural Information Processing Systems - 11*, pages 952–958. MIT Press, 1999.

[23] D. H. Wolpert, K. Wheeler, and K. Tumer. Collective intelligence for control of distributed dynamical systems. *Europhysics Letters*, 49(6), March 2000.

[24] W. Zhang and T. G. Dietterich. Solving combinatorial optimization tasks by reinforcement learning: A general methodology applied to resource-constrained scheduling. *Journal of Artificial Intelligence Reseach*, 2000.