Providing effective access to shared resources: a COIN approach

Stéphane Airiau
Sandip Sen
Mathematical & Computer Sciences Dept
600 South College Avenue
Tulsa, OK 74104
{stephane,sandip}@ens.utulsa.edu

David H. Wolpert
Kagan Tumer
NASA Ames Research Center
MS 269-2
Moffett Field, CA94035
{dhw,kagan}@email.arc.nasa.gov

ABSTRACT

Managers of systems of shared resources typically have many separate goals. Examples are efficient utilization of the resources among its users and ensuring no user's satisfaction in the system falls below a preset minimal level. Since such goals will usually conflict with one another, either implicitly or explicitly the manager must determine the relative importance of the goals, encapsulating that into an overall utility function rating the possible behaviors of the entire system. Here we demonstrate a distributed, robust, and adaptive way to optimize that overall function. Our approach is to interpose adaptive agents between each user and the system, where each such agent is working to maximize its own private utility function. In turn, each such agent's function should be both relatively easy for the agent to learn to optimize, and "aligned" with the overall utility function of the system manager — an overall function that is based on but in general different from the satisfaction functions of the individual users. To ensure this we enhance the COllective INtelligence (COIN) framework to incorporate user satisfaction functions in the overall utility function of the system manager and accordingly in the associated private utility functions assigned to the users' agents. We present experimental evaluations of different COIN-based private utility functions and demonstrate that those COIN-based functions outperform some natural alternatives.

1. INTRODUCTION

One of the key problems confronting the designers of large-scale, distributed agent applications is control of access to shared resources. In order for the system to function well, access to these shared resources must be coordinated to eliminate potential problems like deadlock, starvation, livelock and other forms of resource contention. Without proper coordination, both individual user satisfaction and overall system performance can be significantly impaired.

As hand-coded, static procedures are likely to be brittle in complex environments, we are interested in flexible, adaptive, scalable approaches to the resource sharing problem [2]. One approach is to use distributed machine learning-based agents each of which works exclusively to increase the satisfaction of its associated user. While this is appealing, individual learners optimizing the "satisfaction" utility functions of their users typically will not coordinate their activities and may even work at cross-purposes, thereby degrading the performance of the entire system. Well-known cases of this problem of global shared resources include the Tragedy of the Commons [4]. Such problems highlight the need for careful design of the utility functions each of the learning agents work to optimize. In general, this means having each agent use a utility function that differs from the satisfaction function of its associated user [11]. In addition to ensuring that the private utility functions of the agents do not induce them to work at cross-purposes though, it is also necessary that those functions can be readily optimized in an adaptive manner, despite high noise levels in the environment.

The field of mechanism design, originating in game theory and economics, appears to address this problem. However it suffers from restrictions that diminish its suitability for problems — like the ones considered here — involving bounded-rational, non-human, noisy agents, where the variables controlled by the agents and even the number of agents can be varied [15]. In contrast, the COllective INtelligence (COIN) framework is explicitly formulated to address such problems without such restrictions [16, 11, 14]. In particular, its mathematics derives private utility functions to provide the individual learning agents that are *learnable*, in addition to inducing the agents not to work at cross-purposes. Such functions are designed both so that the agents can perform well at maximizing them, and so that as they do this the agents also "unintentionally" improve the provided "world utility" function rating behavior of the entire system.

In the past, COIN techniques have been used for world utility functions that are fully exogenous, in that they do not reflect any satisfaction functions of a set of users of the system nor how best to accommodate those satisfaction functions. While achieving far superior performance in the domains tested to date than do conventional approaches like greedy agents and team games [10, 16, 13], these tests do not assess how well COIN-based systems perform in situations like shared resource problems, in which the world utility is not fully exogenous. In this paper we start by reviewing the mathematics of collective intelligence. We then test the techniques recommended by that mathematics on shared resource problems, using a world utility function that reflects the satisfaction levels of the individual users of the system, and in particular trades off concern that no individual's satisfaction be too low with concern that the overall system behave efficiently. We demonstrate that in such domains COIN techniques also far outperform approaches like team games. We also validate some quantitative predictions of the COIN mathematics concerning how the world utility is affected by various modifications to the COIN techniques.

2. BACKGROUND ON COLLECTIVE INTELLIGENCE

The "COllective INtelligence" framework (COIN) concerns the design of large distributed collectives of self-interested agents where there exists a world utility function measuring the performance of the entire collective. The emphasis of the research is on the inverse problem: given a set of self-interested agents and a world utility function, how should a designer configure the system, and in particular set the private utility functions of the agents so that, when all agents try to optimize their functions, the entire collective performs better. In this section, we introduce the mathematics of designing collectives. Because of space limitations, the concepts cannot be justified or elaborated in any detail; references to other papers with such details are provided.

2.1 Central Equation

Let ζ be an arbitrary space whose elements z give the joint move of all agents in the collective system. We wish to search for the z that maximizes the provided **world utility** function $\mathcal{G}(z)$. In addition to $\mathcal{G}$ we are concerned with private utility functions $\{g_\eta\}$, one such function for each agent η controlling z_η. We use the notation $\hat{}\eta$ to refer to all agents other than η.

We need a way to "calibrate" the utility functions so that the value assigned to a joint action z reflects the ranking of z relative to all the other possible joint actions in ζ. We denote as intelligence the result of this calibration process. The intelligence can be considered to be the percentile of states that would have resulted in having a worse utility (if 99% of the available actions lead to a worse utility, the agent made a smart decision). In the COIN framework, the "intelligence for η at z with respect to U" is defined by:

$$N_{\eta,U}(z) \equiv \int d\mu_{z_{\hat{}\eta}}(z')\Theta[U(z) - U(z')] \; , \tag{1}$$

where Θ is the Heaviside function [1], and where the subscript on the (normalized) measure $d\mu$ indicates it is restricted to z' sharing the same non-η components as z. There is no particular constraint on the measure μ other than reflecting the type of system (whether ζ is countable or not, if not, what coordinate system is being used...).

As system designer, our uncertainty concerning the state of the system is reflected in a probability distribution P over ζ. Our ability to control the system consists of setting the value of some characteristic of the collective, which is denoted as its **design coordinate**. For instance, the design coordinate can be setting the private utility functions of the agents. For a value s of the design coordinate, our analysis revolves around the following **central equation** for $P(\mathcal{G} \mid s)$, which follows from Bayes' theorem:

$$P(\mathcal{G} \mid s) = \tag{2}$$
$$\underbrace{\int d\vec{N}_{\mathcal{G}} P(\mathcal{G} \mid \vec{N}_{\mathcal{G}}, s)}_{\text{explore vs. exploit}} \int \underbrace{d\vec{N}_g P(\vec{N}_{\mathcal{G}} \mid \vec{N}_g, s)}_{\text{factoredness}} \; \underbrace{P(\vec{N}_g \mid s)}_{\text{learnability}} \; ,$$

where $\vec{N}_g$ and $\vec{N}_{\mathcal{G}}$ are the **intelligence** vectors of the agents with respect to the private utility g_η of η and the world utility $\mathcal{G}$, respectively.

Note that $N_{\eta,g_\eta}(z) = 1$ means that agent η is fully rational at z, in that its move maximizes the value of its utility, given the moves of the agents. In other words, a point z where $N_{\eta,g_\eta}(z) = 1$ for all agents η is one that meets the definition of a game-theory Nash equilibrium. On the other hand, a z at which all components of $\vec{N}_{\mathcal{G}} = 1$ is a maximum $\mathcal{G}$ along all coordinates of z. So if we can get these two points to be identical, then if the agents do well enough at maximizing their private utilities we are assured we will be near an axis-maximizing point for $\mathcal{G}$.

To formalize this, consider our decomposition of $P(\mathcal{G} \mid s)$.

- learnability: if we can choose the global coordinate s so that the third conditional probability in the integrand is peaked around vectors $\vec{N}_g$ all of whose components are close to 1 (that is agents are able to learn their tasks), then we have likely induced large (private utility function) intelligences. Intuitively, this ensures that the private utility functions have high "signal-to-noise".
- factoredness: by choosing s, if we can also have the second term be peaked about $\vec{N}_{\mathcal{G}}$ equal to $\vec{N}_g$ (that is the private utility and the world utility are aligned), then $\vec{N}_{\mathcal{G}}$ will also be large. It is in the second term that the requirement that the private utility functions be "aligned with $\mathcal{G}$" arises. Note that our desired form for the second term in Equation 2 is assured if we have chosen private utilities such that $\vec{N}_g$ equals $\vec{N}_{\mathcal{G}}$ exactly for all z. Such a system is said to be **factored**.
- Finally, if the first term in the integrand is peaked about high $\mathcal{G}$ when $\vec{N}_{\mathcal{G}}$ is large, then our choice of s will likely result in high $\mathcal{G}$, as desired.

On the other hand, unless one is careful, each agent will have a hard time discerning the effect of its behavior on its private utility when the system is large. Typically, each agent has a horrible "signal-to-noise" problem in such a situation. This will result in a poor term three in the central

[1] The Heaviside function is defined to equal 1 when its argument is greater or equal to 0, and to 0 otherwise.

equation. For instance, consider a collective provided by the human economy. A team game in that example would mean that every citizen gets the national GDP as her/his reward signal, and tries to discern how best to act to maximize that reward signal. At the risk of understatement, this would provide the individual members of the economy with a difficult reinforcement learning task.

In this paper, we concentrate on the second and third terms, and show how to simultaneously set them to have the desired form in the next section. Hence, the research problem is to choose s so that the system both has good learnabilities for its agents, and is factored.

Mechanism design might, at first glance, appear to provide us techniques for solving this version of the inverse problem. However while it can be viewed as addressing the second term, the issue of learnability is not studied in mechanism design. Rather mechanism design is almost exclusively concerned with collectives that are at (a suitable refinement of) an exact Nash equilibrium [6]. That means that every agent is assumed to be performing *as well as is theoretically possible*, given the behavior of the rest of the system. In setting private utilities and the like on this basis, mechanism design ignores completely the issue of how to design the system so that each of the agents can achieve a good value of its private utility (given the behavior of the rest of the system). In particular it ignores all statistical issues related to how well the agents can be expected to perform for various candidate private utilities.

Such issues become crucial as one moves to large systems, where each agent is implicitly confronted with a very high-dimensional reinforcement learning task. It is its ignoring of this issue that means that mechanism design scales poorly to large problems. In contrast, the COIN approach is precisely designed to address both learnability issues as well as term 2.

2.2 Wonderful Life Utility and Aristocrat Utility

As an example of the foregoing, any "team game" in which all private utility functions equal $\mathcal{G}$ is factored [1]. However as illustrated above in the human economy example, team games often have very poor forms for term 3 in Equation 2, forms which get progressively worse as the size of the collective grows. This is because for such private utility functions each agent η will usually confront a very poor "signal-to-noise" ratio in trying to discern how its actions affect its utility $g_\eta = \mathcal{G}$, since so many other agent's actions also affect $\mathcal{G}$ and therefore dilute η's effect on its own private utility function.

We now focus on algorithms based on private utility functions $\{g_\eta\}$ that optimize the signal/noise ratio reflected in the third term of the central equation, subject to the requirement that the system be factored. We will introduce two utility functions satisficing this criteria, namely the Wonderful Life Utility (WLU) and the Aristocrat Utility (AU)

To understand how these algorithms work, say we are given an arbitrary function $f(z_\eta)$ over agent η's moves, two such moves ${z_\eta}^1$ and ${z_\eta}^2$, a utility U, a value s of the design coordinate, and a move by all agents other than η, $z_{\hat{\eta}}$. Define the associated **learnability** by

$$\Lambda_f(U; z_{\hat{\eta}}, s, {z_\eta}^1, {z_\eta}^2) \equiv \sqrt{\frac{[E(U; z_{\hat{\eta}}, {z_\eta}^1) - E(U; z_{\hat{\eta}}, {z_\eta}^2)]^2}{\int dz_\eta [f(z_\eta) Var(U; z_{\hat{\eta}}, z_\eta)]}} \ . \tag{3}$$

The expectation values in the numerator are formed by averaging over the training set of the learning algorithm used by agent η, n_η. Those two averages are evaluated according to the two distributions $P(U|n_\eta)P(n_\eta|z_{\hat{\eta}}, {z_\eta}^1)$ and $P(U|n_\eta)P(n_\eta|z_{\hat{\eta}}, {z_\eta}^2)$, respectively. (That is the meaning of the semicolon notation.) Similarly the variance being averaged in the denominator is over n_η according to the distribution $P(U|n_\eta)P(n_\eta|z_{\hat{\eta}}, z_\eta)$.

The denominator in Equation 4 reflects how sensitive $U(z)$ is to changing $z_{\hat{\eta}}$. In contrast, the numerator reflects how sensitive $U(z)$ is to changing z_η. So the greater the learnability of a private utility function g_η, the more $g_\eta(z)$ depends only on the move of agent η, i.e., the better the associated signal-to-noise ratio for η. Intuitively then, so long as it does not come at the expense of decreasing the signal, increasing the signal-to-noise ratio specified in the learnability will make it easier for η to achieve a large value of its intelligence. This can be established formally: if appropriately scaled, g'_η will result in better expected intelligence for agent η than will g_η whenever $\Lambda_f(g'_\eta; z_{\hat{\eta}}, s, {z_\eta}^1, {z_\eta}^2) > \Lambda_f(g_\eta; z_{\hat{\eta}}, s, {z_\eta}^1, {z_\eta}^2)$ for all pairs of moves ${z_\eta}^1, {z_\eta}^2$[14].

One can solve for the set of all private utilities that are factored with respect to a particular world utility. Unfortunately though, in general a collective cannot both be factored and have infinite learnability for all of its agents [14]. However consider **difference** utilities, of the form

$$U(z) = \beta[\mathcal{G}(z) - D(z_{\hat{\eta}})] \tag{4}$$

Any difference utility is factored [14]. In addition, for all pairs ${z_\eta}^1, {z_\eta}^2$, under benign approximations, the difference utility maximizing $\Lambda_f(U; z_{\hat{\eta}}, s, {z_\eta}^1, {z_\eta}^2)$ is found by choosing

$$D(z_{\hat{\eta}}) = E_f(\mathcal{G}(z) \mid z_{\hat{\eta}}, s) \ , \tag{5}$$

up to an overall additive constant, where the expectation value is over z_η. We call the resultant difference utility the **Aristocrat** utility (AU), loosely reflecting the fact that it measures the difference between a agent's actual action and the average action. If each agent η uses an appropriately rescaled version of the associated AU as its private utility function, then we have ensured good form for both terms 2 and 3 in Equation 2.

Using AU in practice is sometimes difficult, due to the need to evaluate the expectation value. Fortunately there are other utility functions that, while being easier to evaluate than AU, still are both factored and possess superior learnability to the team game utility, $g_\eta = \mathcal{G}$. One such private utility 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 possible moves:

$$WLU_\eta \equiv \mathcal{G}(z) - \mathcal{G}(z_{\hat{\eta}}, CL_\eta) \ . \tag{6}$$

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, and therefore (when appropriately scaled) results in better expected intelligence [11, 16, 14].

3. PROBLEM DEFINITION

Let consider a set of users and a set of m shared resources/machines. Different users have different task loads, each task being of one of T types. Any machine can perform any task type, but different machines have different processing speeds for each task types. A given user sends all of its tasks of a particular type to a specific machine but can send tasks of different types to different machines. So each user must decide, for each $j \in T$, to what machine A_i^j to send all of its tasks of type j.

We consider a batch mode scenario, in which every user submits all of its tasks, the machines complete their tasks, and the associated overall performance of the system is ascertained. Each machine works by grouping all the tasks sent to it by type, and performs all tasks of one type contiguously before returning them to the associated users and then switching to the next type. The order of processing different task types is fixed for any given machine. Each type is performed at a speed specific to the machine.

3.1 Satisfaction functions of users

User i has a personal "satisfaction" utility function, $\mathcal{H}_i$, which is an inverse function of the time that the user has to wait for all of his tasks to finish. The user may be less willing to wait for certain task types compared to others. The user can provide feedback or his model of satisfaction to the system manager.

We indicate the completion time of the tasks of type j submitted by user i to machine A_i^j by CT_i^j.

The functions defined below measure dissatisfaction rather than satisfaction, in that they are monotonically increasing functions of the delay. So the goal for agent i is to maximize $\mathcal{H}_i = -\mathcal{D}_i$, where $\mathcal{D}_i$ is defined as one of the following:
• the maximum delay for user i, $\mathcal{D}_i = \max_{j \in \{1..T\}} CT_i^j$,
• an importance-weighted combination of the time of completion of all the tasks of the user, $\mathcal{D}_i = \sum_{j=1}^{T} \alpha_j CT_i^j$.
$\mathcal{D} \equiv \{\mathcal{D}_i\}$ is the set of the dissatisfactions of all users.

3.2 Definition of the World Utility Function $\mathcal{G}$

The world utility function measures the performance of the overall system. It is the responsibility of the system designer to set this function. He might decide to measure performance by measuring some characteristics of the use of the resource only (e.g. the load distribution or the idle time). The system designer might also incorporate user preferences to measure the performance of the system. In our simulations, we have focused on that aspect by defining the world utility function as a function of the dissatisfactions of all users. This assumes that either the users have provided a model of their preferences to the system designer or that the system designer has modeled them. In the former case, users may be allowed to update their model.

We have experimented with the following world utilities:

• Avoid hurting any user: minimize $\mathcal{G}(\mathcal{D}_i) = \max_i \mathcal{D}_i$.

• Minimize a linear combination of the satisfaction of individual users where the weight associated with a user represents the importance of that particular user: $\mathcal{G}(\mathcal{D}_i) = \sum_i w_i \mathcal{D}_i$.

• To have high average satisfaction without any user's satisfaction being low, minimize $\mathcal{G}(\mathcal{D}_i) = \beta * \sigma_{\mathcal{D}} + \sum_i w_i \mathcal{D}_i$, $\sigma_{\mathcal{D}}$ being the variance in the dissatisfactions.

3.3 Agent learning algorithms

Once the measure of dissatisfaction of each agent and the performance metric of the overall system have been defined, the remaining question is how to improve the performance. In the experiments, we want to answer the question: what is the function that each self interested agent has to improve? We refer to this function as the **private utility** for the agent. In other words, given the agents' satisfaction model and the performance metric of the system, we want to be able to tell the agents that the best way to improve the performance of the system (which include their own preferences) is to optimize a certain private utility. If they decide to optimize other criteria, the performance of the system will not be as good as if they follow the recommendation.

We compared performance ensuring from four private utility functions for the agents, the last two being those recommended by the COIN framework:
• a team game; each agent's utility equals $\mathcal{G}$, i.e. by choosing its action, each agent is trying to optimize the performance of the overall system (the agent only focuses on the result of the team, it does not consider its own performance);
• a **greedy** game: each agent i's private utility is $\mathcal{H}_i$ (the agent focuses only on its performance);
• AU; to compute AU for an agent $i \in U$, we need to evaluate $\mathcal{G} - \sum_{j=1}^{T} \mathcal{G}(i,j)$ where $\mathcal{G}(i,j)$ denotes the world utility when agent i sends its job to machine j while all other agents send their jobs to the same machines they actually used. Thus, we need to re-evaluate the completion times of all machines m when the other agents maintain their decision while agent i sending its load to m). See [14] for a discussion of efficient evaluation of AU;
• WLU; to compute WLU for agent i, we chose to clamp to "null" the action of i. Hence, $WLU_i = \mathcal{G} - \mathcal{G}(i, \emptyset)$. Thus we only need to evaluate what the completion time of the machine chosen by i would be if η did not send any load to that machine. See [14] for a discussion of efficient evaluation of AU.

We had each agent use an extremely simple reinforcement learning algorithm in our experiments, so that performance more accurately reflects the quality of the agents' utility functions rather than the sophistication of the learning scheme. In this paper, each agent is "blind", knowing nothing about the environment in which it operates, and observing only the utility value it gets after an iteration of the batch process. The agents each used a modified version a *Boltzmann learning algorithm* to map the set of such utility values to a probability distribution over the (finite) set of possible moves, a distribution that is then sampled to select the agent's move for the next iteration. Each agent tries to learn the estimated reward $\bar{R}_i$ corresponding to the action i. An agent will choose the action i with probability $\mathcal{P}_i = \frac{e^{-\beta \bar{R}_i}}{\sum_{\text{all actions j}} e^{-\beta \bar{R}_j}}$. The parameter β is the *inverse temperature*. During round r, if the action i is chosen, the update of the estimated reward for action i is $\bar{R}_i = \sum_{j \in \mathcal{C}_i} \frac{e^{-\alpha(r-j)}}{\sum_{k \in \mathcal{C}_i} e^{-\alpha(r-k)}} R_j$, where $\mathcal{C}_i$ denotes the set of rounds where action i was chosen, and R_j denotes the reward received at the end of the round j. The *data aging* parameter α models the importance of the choices made in the previous rounds. In our modification, rather than have the expected utility value for each possible move of an agent be a uniform average of the utilities that arose in the past when it made that move, we exponentially discount moves made further into the past, to reflect the non-stationarity of the system.

4. EXPERIMENTS

We experiment with a domain where users are sending different types of printing jobs to a shared pool of heterogeneous printers. The different types of printing jobs may represent different kind of printing operations (for instance, printing black & white, printing in color) Though each printer is able to process any job type, a given printer processes jobs of different types at different speeds. For each printer, we randomly choose the speed of processing each task type, and processing occurs in the order of decreasing speed.

Each user has a load of printing jobs to perform. We assign one agent to handle all tasks of a given type for a given user, and its job is to select a printer to which these tasks would be sent. Each run starts with an initial sequence of iterations in which the agents make purely random moves to generate data for the learning algorithms. We then successively "turn on" more and more of those algorithms i.e., at each successive iteration, a few more agents start to make their moves based on their learning algorithm and associated data rather than randomly. We start each agent with a high *Boltzmann temperature*, usually 10, and decrease it at each iteration by multiplying it by a decay factor. We refer to a temperature schedule as *slow* when the decay factor is large (for instance 99%), and refer to the schedule as *fast* when the decay factor is small (for instance 70%). We use an aging parameter in forming expectation values of 0.1.

In the experiments reported here, the dissatisfaction of a given user is $\mathcal{D}_i = \sum_{j=1}^{T} \alpha_j CT_i^j$ where the α_j are randomly chosen and remain the same throughout the learning process. We used 10 users, 8 types and 6 machines.

4.1 Results

We first considered the case where the World Utility Function $\mathcal{G}$ is computed as the maximum of the dissatisfaction of the user, i.e., $\mathcal{G} = \max_i \mathcal{D}_i$. The goal is to minimize the maximum dissatisfaction among the users. The decay factor used is 0.99. In Fig. 1 we provide performance for the different agent utility functions averaged over 10 runs. The performances achieved by AU and WLU are similar, and dominate that of both Team Game and Greedy agent utilities. We found that at the end of the learning process, the standard deviation in $\mathcal{G}$ for AU and WLU is small (respectively 2.1 and 1.08) whereas it is large for Greedy and Team Game (respectively 14.4 and 24). This corroborates the reasoning that signal to noise is too large when we use Greedy or Team Game, and therefore the learning agents are not able to determine the impact of their actions on their utility functions. In contrast, as discussed above, AU and WLU are designed to have good signal to noise, which enables them to reach better (and more consistent) performance levels.

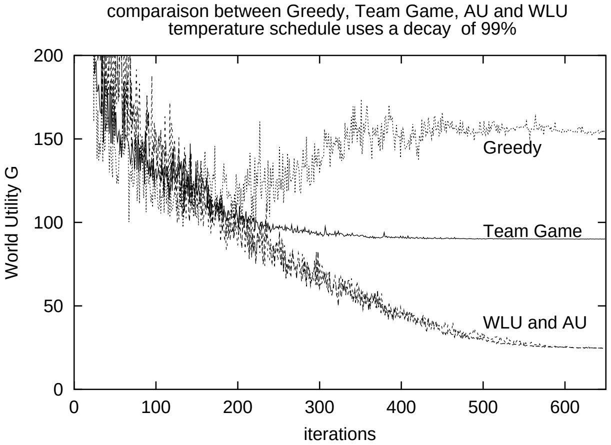

Figure 1: Comparison between different agent utilities when $\mathcal{G} = \max_i \mathcal{D}_i$

Note that with the greedy approach $\mathcal{G}$ *worsens* as each agent learns how to maximize its utility. This is an example of a tragedy of the commons: since the Greedy utility is not factored with respect to $\mathcal{G}$, the associated Nash equilibrium — achieved as the agents learn more — has poor $\mathcal{G}$.

In Fig. 2, $\mathcal{G}$ is instead the weighted sum of the dissatisfaction of the users, i.e., $\mathcal{G} = \sum_i w_i \mathcal{D}_i$. In figure 3 it is the sum of all the users dissatisfactions with the standard deviation of those dissatisfactions, i.e., $\mathcal{G} = \beta * \sigma_{\mathcal{D}} + \sum_i \mathcal{D}_i$ (we have used $\beta = 1$). The decay factor used in these experiments is 0.99. In all cases, AU and WLU achieve comparable performance, and outperform team game.

Although the asymptotic performance of the team game is not as good as that of AU or WLU, in both of these experiments team game performs better than AU or WLU in the early phase of the learning. In addition the convergence in time of $\mathcal{G}$ for AU and WLU is slower in these experiments. This is consistent with COIN theory, which says that simply changing to a more learnable utility function may actually lead to *worse* performance if the learning temperature is not changed simultaneously. Intuitively, to achieve their better signal to noise ratios, AU and WLU shrink both the signal and the noise, and therefore need both to be rescaled upward. In the context of Boltzmann learning algorithms, this is equivalent to lowering their temperatures.

To investigate this issue, in a second set of experiments we used different decay factors and faster schedules. In

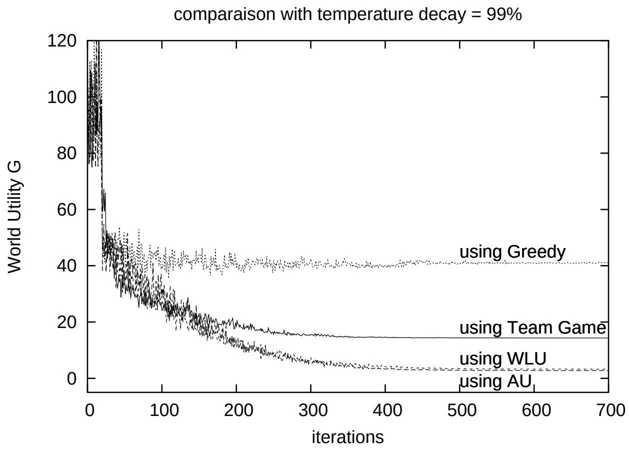

Figure 2: Comparison between different reward in the case where $\mathcal{G} = \sum_i w_i \mathcal{D}_i$.

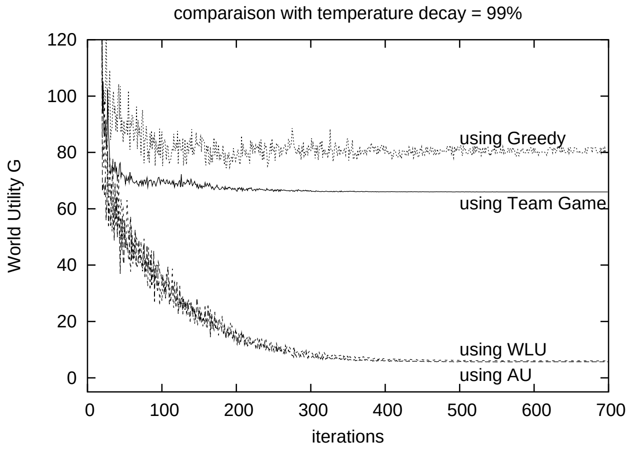

Figure 3: Performances for $\mathcal{G} = \sigma + \sum_i \mathcal{D}_i$.

Fig. 4 we present the asymptotic world utility value for Team Game, AU, and WLU for different decay factors and starting temperatures. In the experiments plotted we used $\mathcal{G} = \sum_i w_i \mathcal{D}_i$, but we observed the same phenomena with the other $\mathcal{G}$ discussed above: the team game never outperforms AU or WLU. Moreover, if for each time-algorithm pair we used the schedule that is optimal for that pair, then is no time t at which the performance of the team game is better than the performance of AU/WLU. Fig. 5 illustrates this, showing that with a fast schedule having a decay factor of 0.65, AU and WLU converge faster than team game.

In addition to these advantages, we found that the performance with the team game is more sensitive to the change in temperature schedule, again in accord with COIN theory. Indeed, although convergence to asymptotia is faster for all agent utilities with the faster schedule (see Fig. 5), from Fig. 4 we see that both average asymptotic performance and its standard deviation are substantially for the team game for the faster schedule. In contrast, AU and WLU do not suffer as much from the switch to a faster schedule.

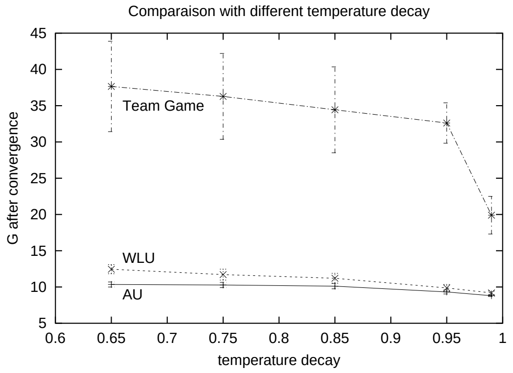

Figure 4: Different decay factors ($\mathcal{G} = \sum_i w_i \mathcal{D}_i$).

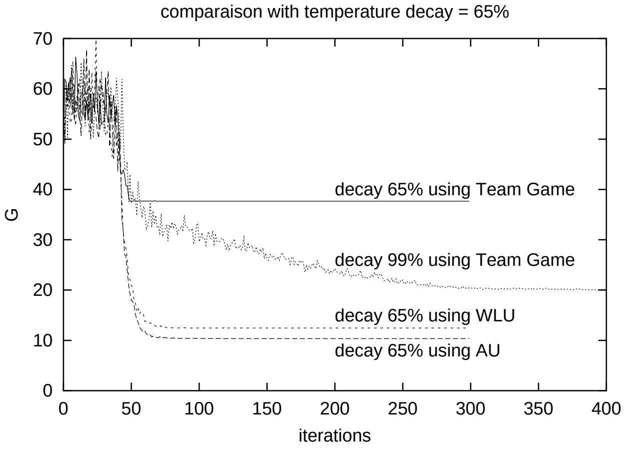

Figure 5: Performance for $\mathcal{G} = \sum_i w_i \mathcal{D}_i$.

We also investigated the scaling properties of these approaches as one increases the size of the system and found that AU and WLU scale up much better than the team game approach, again in accord with COIN theory. Intuitively, the larger the system, the worse the signal-to-noise problems with utilities like team games, and therefore the greater the gain in using a utility like AU or WLU that corrects for it. In Fig. 6, we used a fixed number of types, 8, and a fixed ratio of 1 machine for 5 users. We increased the number of users from 5 to 25. For each data point, we considered a few scenarios (different machine configurations, different loads distributed to users, etc.) and performed 10 runs for each scenario. Each point represents the average over the scenarios. The use of AU and WLU yield comparable results, and the difference in performance between them and Team Game increases with the number of agents.

Since in general one cannot compute the best possible $\mathcal{G}$ value, we cannot make absolute performance claims. Nonetheless, both in terms of performance and scaling, use of WLU or AU is clearly preferable to a team game or greedy approach, for several different choices for $\mathcal{G}$.

5. RELATED WORK

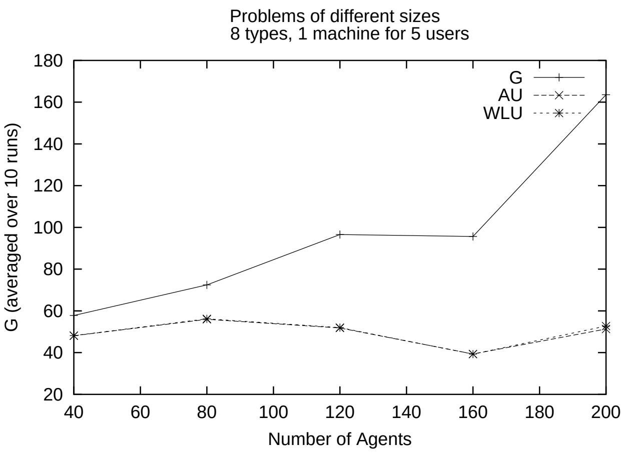

Figure 6: Scaling properties for $\mathcal{G} = \sum_i \mathcal{D}_i$.

Multiagent system researchers have studied balancing of load across shared resources with different decision-making procedures. These approaches include the following:

- Studying chaotic nature of resource loads when each agent uses a greedy selection procedures [5].
- Effect of limited knowledge on system stability [7, 9].
- Market mechanisms for optimizing quality-of-service [17].
- Using reinforcement learning to balance loads [8].
- Social dilemma problems that arise when individual agents try to greedily exploit shared resources [3, 11].
- Distinguishing easy vs. difficult resource allocation [12].

The typical assumption in most of this work is that each user has an atomic load to send to one of several equivalent resources. Our work addresses a more general scenario where a user has multiple task loads and resources are heterogeneous. So not only may the decisions of the users conflict, but decisions for different tasks taken by the same user can interfere with each other. Also, we explicitly handle the issue of user satisfaction metrics (something awkward to incorporate in load-balancing, for example) and variability in the world utility, and use the COIN procedure to ensure that the combination of individual user satisfaction metrics are optimized. The combination can include weights for different users and also, if necessary, reduce disparate levels of satisfaction from the outcome by minimizing the standard deviation of satisfaction. With the COIN procedure, all of this is done using very simple agents, in a highly parallelizable fashion, with no modeling of the underlying dynamics of the system.

6. DISCUSSION

The COIN framework has been applied to many domains, including domains requiring sequence of actions. In these cases, the world utility function was decided first, and then the local utility function of each agent was derived from it. This paper differs in two main ways from the COIN applications studied so far. First, agents have preferences represented by a dissatisfaction function. In previous work, the individual in the system did not have any intrinsic preferences. Secondly, the world utility is based upon the preferences of users and a system administrator. The main contribution of the paper is to show that the users, in order to achieve satisficing performance of the overall system, had better prefer to optimize COIN based private utilities. The results demonstrate that even in the case where the world utility function is not exogenous, the previously superiority of COIN technique holds. In particular, the utilities AU and WLU outperform locally greedy approaches and more importantly, the Team Game approach, in terms of average performance, variability in performance, and robustness against changes to the parameters of the agent's learning algorithm. Moreover, these improvements grow dramatically as the system size is increased, an extremely important consideration in future applications.

We are currently experimenting with aggregating several printing decisions under the jurisdiction of one agent. This would reduce the number of agents in the system, and possibly both speed up the convergence of the performance and improve it. In particular we will investigate different kind of aggregations, e.g., based on users, based on task types, crossing both, etc. There is also the interesting possibility of learning the best aggregation of decisions into individual agents.

7. REFERENCES

[1] 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.

[2] E. H. Durfee. Scaling up coordination strategies. *IEEE Computer*, 34(7):39–46, July 2001.

[3] N. S. Glance and T. Hogg. Dilemmas in computational societies. In *First International Conference on Multiagent Systems*, pages 117–124, Menlo Park, CA, 1995. AAAI Press/MIT Press.

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

[5] J. O. Kephart, T. Hogg, and B. A. Huberman. Dynamics of computational ecosystems: Implications for DAI. In M. N. Huhns and L. Gasser, editors, *Distributed Artificial Intelligence*, volume 2 of *Research Notes in Artificial Intelligence*. Pitman, 1989.

[6] D. C. Parkes. *iterative combinatorial auctions: Theory and practice, 2001.*

[7] S. K. Rustogi and M. P. Singh. Be patient and tolerate imprecision: How autonomous agents can coordinate effectively. In Proceedings of the International Joint Conference on Artificial Intelligence*, pages 512–517, 1999.*

[8] A. Schaerf, Y. Shoham, and M. Tennenholtz. Adaptive load balancing: A study in multiagent learning. Journal of Artificial Intelligence Research*, 2:475–500, 1995.*

[9] S. Sen, N. Arora, and S. Roychowdhury. Using limited information to enhance group stability. International Journal of Human-Computer Studies*, 48:69–82, 1998.*

[10] K. Tumer, A. K. Agogino, and D. H. Wolpert. Learning sequences of actions in collectives of autonomous agents. In Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 378–385, New York: NY, 2002. ACM Press.*

[11] K. Tumer and D. H. Wolpert. Collective intelligence and braess' paradox. In Proceedings of the Seventeenth National Conference on Artificial Intelligence*, pages 104–109, Menlo Park, CA, 2000. AAAI Press.*

[12] H. van Dyke Parunak, S. Brueckner, J. Sauter, and R. Savit. Effort profiles in multi-agent resource allocation. In Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 248–255, New York: NY, 2002. ACM Press.*

[13] D. H. Wolpert, S. Kirshner, C. J. Merz, and K. Tumer. Adaptivity in agent-based routing for data networks. In Proc. of the 4th International Conference on Autonomous Agents*, pages 396–403, 2000.*

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

[15] D. H. Wolpert and K. Tumer. Beyond mechanism design. In H. G. et al., editor, Proceedings of International Congress of Mathematicians*. Qingdao Publishing, 2002.*

[16] D. H. Wolpert, K. R. Wheeler, and K. Turner. General principles of learning-based multi-agent systems. In Proceedings of the Third International Conference on Autonomous Agents*, pages 77–83, New York: NY, 1999. ACM Press.*

[17] H. Yamaki, Y. Yamauchi, and T. Ishida. Implementation issues on market-based qos control. In Proceedings of the Third International Conference on Multi-Agent Systems*, pages 357–364, 1998.*