Sharing Experiences to Learn User Characteristics in Dynamic Environments with Sparse Data

David Sarne, Barbara J. Grosz
School of Engineering and Applied Sciences
Harvard University, Cambridge MA 02138 USA
{sarned,grosz}@eecs.harvard.edu

ABSTRACT

This paper investigates the problem of estimating the value of probabilistic parameters needed for decision making in environments in which an agent, operating within a multi-agent system, has no a priori information about the structure of the distribution of parameter values. The agent must be able to produce estimations even when it may have made only a small number of direct observations, and thus it must be able to operate with sparse data. The paper describes a mechanism that enables the agent to significantly improve its estimation by augmenting its direct observations with those obtained by other agents with which it is coordinating. To avoid undesirable bias in relatively heterogeneous environments while effectively using relevant data to improve its estimations, the mechanism weighs the contributions of other agents' observations based on a real-time estimation of the level of similarity between each of these agents and itself. The "coordination autonomy" module of a coordination-manager system provided an empirical setting for evaluation. Simulation-based evaluations demonstrated that the proposed mechanism outperforms estimations based exclusively on an agent's own observations as well as estimations based on an unweighted aggregate of all other agents' observations.

Categories and Subject Descriptors

I.2.6 [**Artificial Intelligence**]: Learning—*Parameter learning*; I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*Intelligent agents, Multiagent systems*; G.3 [**Mathematics of Computing**]: Probability and Statistics—*Distribution functions*

General Terms

Algorithms, Experimentation

Keywords

Adjustable autonomy, Interruption management

1. INTRODUCTION

For many real-world scenarios, autonomous agents need to operate in dynamic, uncertain environments in which they have only incomplete information about the results of their actions and characteristics of other agents or people with whom they need to cooperate or collaborate. In such environments, agents can benefit from sharing information they gather, pooling their individual experiences to improve their estimations of unknown parameters required for reasoning about actions under uncertainty.

This paper addresses the problem of learning the distribution of the values of a probabilistic parameter that represents a characteristic of a person who is interacting with a computer agent. The characteristic to be learned is (or is clearly related to) an important factor in the agent's decision making.[1] The basic setting we consider is one in which an agent accumulates observations about a specific user characteristic and uses them to produce a timely estimate of some measure that depends on that characteristic's distribution. The mechanisms we develop are designed to be useful in a range of application domains, such as disaster rescue, that are characterized by environments in which conditions may be rapidly changing, actions (whether of autonomous agents or of people) and the overall operations occur at a fast pace, and decisions must be made within tightly constrained time frames. Typically, agents must make decisions in real time, concurrent with task execution, and in the midst of great uncertainty. In the remainder of this paper, we use the term "fast-paced" to refer to such environments. In fast-paced environments, information gathering may be limited, and it is not possible to learn offline or to wait until large amounts of data are collected before making decisions.

Fast-paced environments impose three constraints on any mechanism for learning a distribution function (including the large range of Bayesian update techniques [23]): (a) the *no structure constraint:* no a priori information about the structure of the estimated parameter's distribution nor any initial data from which such structure can be inferred is available; (b) the *limited use constraint:* agents typically need to produce only a small number of estimations in total for this parameter; (c) the *early use constraint:* high accuracy is a critical requirement even in the initial stages of learning. Thus, the goal of the estimation methods presented in this paper is to minimize the average error over time, rather than to determine an accurate value at the end of a long period of interaction. That is, the agent is expected to work with the user for a limited time, and it attempts to minimize the overall error in its estimations. In such environments, an agent's individually acquired data (its own observations) are too sparse for it to obtain good estimations in the requisite time frame. Given the no-structure-constraint of the environment, approaches that depend on structured distributions may result in a significantly high estimation bias.

We consider this problem in the context of a multi-agent distributed system in which computer agents support people who are carrying out complex tasks in a dynamic environment. The fact that agents are part of a multi-agent setting, in which other agents may

[1] Learning the distribution rather than just determining some value in the distribution is important whenever the overall shape of the distribution and not just such individual features as mean are important.

AAMAS'07, May 14–18, 2007, Honolulu, Hawai'i, USA.

978-81-904262-7-5 (RPS)

also be gathering data to estimate a similar characteristic of their users, offers the possibility for an agent to augment its own observations with those of other agents, thus improving the accuracy of its learning process. Furthermore, in the environments we consider, agents are usually accumulating data at a relatively similar rate. Nonetheless, the extent to which the observations of other agents will be useful to a given agent depends on the extent to which their users' characteristics' distributions are correlated with that of this agent's user. There is no guarantee that the distribution for two different agents is highly, positively correlated, let alone that they are the same. Therefore, to use a data-sharing approach, a learning mechanism must be capable of effectively identifying the level of correlation between the data collected by different agents and to weigh shared data depending on the level of correlation.

The design of a *coordination autonomy* (CA) module within a coordination-manager system (as part of the DARPA *Coordinators* project [18]), in which agents support a distributed scheduling task, provided the initial motivation and a conceptual setting for this work. However, the mechanisms themselves are general and can be applied not only to other fast-paced domains, but also in other multi-agent settings in which agents are collecting data that overlaps to some extent, at approximately similar rates, and in which the environment imposes the no-structure, limited- and early-use constraints defined above (e.g., exploration of remote planets). In particular, our techniques would be useful in any setting in which a group of agents undertakes a task in a new environment, with each agent obtaining observations at a similar rate of individual parameters they need for their decision-making.

In this paper, we present a mechanism that was used for learning key user characteristics in fast-paced environments. The mechanism provides relatively accurate estimations within short time frames by augmenting an individual agent's direct observations with observations obtained by other agents with which it is coordinating. In particular, we focus on the related problems of estimating the cost of interrupting a person and estimating the probability that that person will have the information required by the system. Our adaptive approach, which we will refer to throughout the paper as *"selective-sharing"*, allows our CA to improve the accuracy of its distribution-based estimations in comparison to relying only on the interactions with a specific user (subsequently, *"self-learning"*) or pooling all data unconditionally (*"average all"*), in particular when the number of available observations is relatively small.

The mechanism was successfully tested using a system that simulates a Coordinators environment. The next section of the paper describes the problem of estimating user-related parameters in fast-paced domains. Section 3 provides an overview of the methods we developed. The implementation, empirical setting, and results are given in Sections 4 and 5. A comparison with related methods is given in Section 6 and conclusions in section 7.

2. PARAMETER ESTIMATION IN FAST-PACED DOMAINS

The CA module and algorithms we describe in this paper were developed and tested in the Coordinators domain [21]. In this domain, autonomous agents, called "Coordinators", are intended to help maximize an overall team objective by handling changes in the task schedule as conditions of operation change. Each agent operates on behalf of its *owner* (e.g., the team leader of a first-response team or a unit commander) whose schedule it manages. Thus, the actual tasks being scheduled are executed either by owners or by units they oversee, and the agent's responsibility is limited to maintaining the scheduling of these tasks and coordinating with the agents of other human team members (i.e., other owners). In this domain, scheduling information and constraints are distributed. Each agent receives a different view of the tasks and structures that constitute the full multi-agent problem—typically only a partial, local one. Schedule revisions that affect more than one agent must be coordinated, so agents thus must share certain kinds of information. (In a team context they may be designed to share other types as well.) However, the fast-paced nature of the domain constrains the amount of information they can share, precluding a centralized solution; scheduling problems must be solved distributively.

The agent-owner relationship is a collaborative one, with the agent needing to interact with its owner to obtain task and environment information relevant to scheduling. The CA module is responsible for deciding intelligently when and how to interact with the owner for improving the agent's scheduling. As a result, the CA must estimate the expected benefit of any such interaction and the cost associated with it [19]. In general, the net benefit of a potential interaction is $PV - C$, where V is the value of the information the user may have, P is the probability that the user has this information, and C is the cost associated with an interaction. The values of P, V, and C are time-varying, and the CA estimates their value at the intended time of initiating the interaction with its owner. This paper focuses on the twin problems of estimating the parameters P and C, both of which are user-centric in the sense of being determined by characteristics of the owner and the environment in which the owner is operating); it presumes a mechanism for determining V [18].

2.1 Estimating Interruption Costs

The cost of interrupting owners derives from the potential degradation in performance of tasks they are doing caused by the disruption [1; 9, inter alia]. Research on interaction management has deployed sensor-based statistical models of human interruptibility to infer the degree of distraction likely to be caused by an interruption. This work aims to reduce interruption costs by delaying interruptions to times that are convenient. It typically uses Bayesian models to learn a user's current or likely future focus of attention from an ongoing stream of actions. By using sensors to provide continuous incoming indications of the user's attentional state, these models attempt to provide a means for computing probability distributions over a user's attention and intentions [9]. Work which examines such interruptibility-cost factors as user frustration and distractability [10] includes work on the cost of repeatedly bothering the user which takes into account the fact that recent interruptions and difficult questions should carry more weight than interruptions in the distant past or straightforward questions [5].

Although this prior work uses interruptibility estimates to balance the interaction's estimated importance against the degree of distraction likely to be caused, it differs from the fast-paced environments problem we address in three ways that fundamentally change the nature of the problem and hence alter the possible solutions. First, it considers settings in which the computer system has information that may be relevant to its user rather than the user (owner) having information needed by the system, which is the complement of the information exchange situation we consider. Second, the interruptibility-estimation models are task-based. Lastly, it relies on continuous monitoring of a user's activities.

In fast-paced environments, there usually is no single task structure, and some of the activities themselves may have little internal structure. As a result, it is difficult to determine the actual attentional state of agent-owners [15]. In such settings, owners must make complex decisions that typically involve a number of other members of their units, while remaining reactive to events that diverge from expectations [24]. For instance, during disaster rescue, a first-response unit may begin rescuing survivors trapped in a burning house, when a wall collapses suddenly, forcing the unit to

retract and re-plan their actions.

Prior work has tracked users' focus of attention using a range of devices, including those able to monitor gestures [8] and track eye-gaze to identify focus of visual attention [13, 20], thus enabling estimations of cognitive load and physical indicators of performance degradation. The mechanisms described in this paper also presume the existence of such sensors. However, in contrast to prior work, which relies on these devices operating continuously, our mechanism presumes that fast-paced environments only allow for the activation of sensors for short periods of time on an ad hoc basis, because agents' resources are severely limited.

Methods that depend on predicting what a person will do next based only on what the user is currently doing (e.g., MDPs) are not appropriate for modeling focus of attention in fast-paced domains, because an agent cannot rely on a person's attentional state being well structured and monitoring can only be done on a sporadic, non-continuous basis. Thus, at any given time, the cost of interaction with the user is essentially probabilistic, as reflected over a single random monitoring event, and can be assigned a probability distribution function. Consequently, in fast-paced environments, an agent needs a sampling strategy by which the CA samples its owner's interruptibility level (with some cost) and decides whether to initiate an interaction at this specific time or to delay until a lower cost is observed in future samplings. The method we describe in the remainder of this subsection applies concepts from economic search theory [16] to this problem. The CA's cost estimation uses a mechanism that integrates the distribution of an owner's interruptibility level (as estimated by the CA) into an economic search strategy, in a way that the overall combined cost of sensor costs and interaction costs is minimized.

In its most basic form, the economic search problem aims to identify an opportunity that will minimize expected cost or maximize expected utility. The search process itself is associated with a cost, and opportunities (in our case, interruption opportunities) are associated with a stationary distribution function. We use a sequential search strategy [16] in which one observation is drawn at a time, over multiple search stages. The dominating strategy in this model is a reservation-value based strategy which determines a lower bound, and keeps drawing samples as long as no opportunity above the bound was drawn.

In particular, we consider the situation in which an agent's owner has an interruption cost described by a probability distribution function (pdf) $f(x)$ and a cumulative distribution function (cdf) $F(x)$. The agent can activate sensing devices to get an estimation of the interruption cost, x, at the current time, but there is a cost c of operating the sensing devices for a single time unit. The CA module sets a reservation value and as long as the sensor-based observation x is greater than this reservation value, the CA will wait and re-sample the user for a new estimation.

The expected cost, $V(x_{rv})$, using such a strategy with reservation value x_{rv} is described by Equation 1,

$$V(x_{rv}) = \frac{c + \int_{y=0}^{x_{rv}} yf(y)}{F(x_{rv})}, \quad (1)$$

which decomposes into two parts. The first part, c divided by $F(x_{rv})$, represents the expected sampling cost. The second, the integral divided by $F(x_{rv})$, represents the expected cost of interruption, because the expected number of search cycles is (random) geometric and the probability of success is $F(x_{rv})$. Taking the derivative of the left-hand-side of Equation 1 and equating it to zero, yields the characteristics of the optimal reservation value, namely x^*_{rv} must satisfy,

$$V(x^*_{rv}) = x^*_{rv}. \quad (2)$$

Substituting (2) in Equation 1 yields Equation 3 (after integration by parts) from which the optimal reservation value, x^*_{rv}, and consequently (from Equation 2) $V(x^*_{rv})$ can be computed.

$$c = \int_{y=0}^{x^*_{rv}} F(y) \quad (3)$$

This method, which depends on extracting the optimal sequence of sensor-based user sampling, relies heavily on the structure of the distribution function, $f(x)$. However, we need only a portion of the distribution function, namely from the origin to the reservation value. (See Equation 1 and Figure 1.) Thus, when we consider sharing data, it is not necessary to rely on complete similarity in the distribution function of different users. For some parameters, including the user's interruptibility level, it is enough to rely on similarity in the relevant portion of the distribution function. The implementation described in Sections 4-5 relies on this fact.

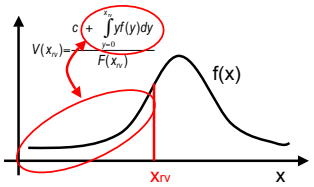

Figure 1: The distribution structure affecting the expected cost's calculation

2.2 Estimating the Probability of Having Information

One way an agent can estimate the probability a user will have information it needs (e.g., will know at a specific interruption time, with some level of reliability, the actual outcome of a task currently being executed) is to rely on prior interactions with this user, calculating the ratio between the number of times the user had the information and the total number of interactions. Alternatively, the agent can attempt to infer this probability from measurable characteristics of the user's behavior, which it can assess without requiring an interruption. This indirect approach, which does not require interrupting the user, is especially useful in fast-paced domains.

The CA module we designed uses such an indirect method: owner-environment interactions are used as a proxy for measuring whether the owner has certain information. For instance, in Coordinators-like scenarios, owners may obtain a variety of information through occasional coordination meetings of all owners, direct communication with other individual owners participating in the execution of a joint task (through which they may learn informally about the existence or status of other actions they are executing), open communications they overhear (e.g. if commanders leave their radios open, they can listen to messages associated with other teams in their area), and other formal or informal communication channels [24]. Thus, owners' levels of communication with others, which can be obtained without interrupting them, provide some indication of the frequency with which they obtain new information. Given occasional updates about its owner's level of communication, the CA can estimate the probability that a random interaction with the owner will yield the information it needs. Denoting the probability distribution function of the amount of communication the user generally maintains with its environment by $g(x)$, and using the transformation function $Z(x)$, mapping from a level of communication, x, to a probability of having the information, the expected probability of getting the information that is needed from the owner when interrupting at a given time can be calculated from

$$P = \int_0^\infty Z(x)g(x)dy. \quad (4)$$

The more observations an agent can accumulate about the distribution of the frequency of an owner's interaction with the environment at a given time, the better it can estimate the probability the owner has the information needed by the system.

3. THE SELECTIVE-SHARING MECHANISM

This section presents the selective-sharing mechanism by which the CA learns the distribution function of a probabilistic parameter by taking advantage of data collected by other CAs in its environment. We first explain the need for increasing the number of observations used as the basis of estimation and then present a method for determining how much data to adopt from other agents.

The most straightforward method for the CA to learn the distribution functions associated with the different parameters characterizing an owner is by building a histogram based on the observations it has accumulated up to the estimation point. Based on this histogram, the CA can estimate the parameter either by taking into account the entire range of values (e.g., to estimate the mean) or a portion of it (e.g., to find the expected cost when using a reservation-value-based strategy). The accuracy of the estimation will vary widely if it is based on only a small number of observations.

For example, Figure 2 illustrates the reservation-value-based cost calculated according to observations received from an owner with a uniform interruption cost distribution $U(0, 100)$ as a function of the number of accumulated observations used for generating the distribution histogram. (In this simulation, device activation cost was taken to be $c = 0.5$).

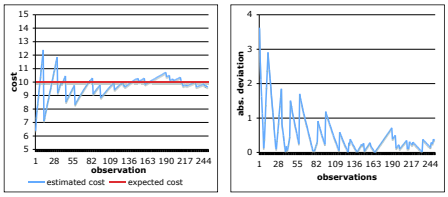

Figure 2: The convergence of a single CA to its optimal strategy

These deviations from the actual (true) value (which is 10 in this case, according to Equation 3) is because the sample used in each stage cannot accurately capture the actual structure of the distribution function. Eventually this method yields a very accurate estimation for the expected interruption cost. However, in the initial stages of the process, its estimation deviates significantly from the true value. This error could seriously degrade the CA's decision making process: underestimating the cost may result in initiating costly, non-beneficial interactions, and overestimating the cost might result in missing opportunities for valuable interactions. Any improvement that can be achieved in predicting the cost values, especially in the initial stages of learning, can make a significant difference in performance, especially because the agent is severely limited in the number of times it can interact with its owner in fast-paced domains.

One way to decrease the deviation from the actual value is by augmenting the data the CA acquires by observing its owner with observations made by other owners' agents. Such an approach depends on identifying other owners with distribution functions for the characteristic of interest similar to the CA's owner. This data-augmentation idea is simple: different owners may exhibit similar basic behaviors or patterns in similar fast-paced task scenarios. Since they are all coordinating on a common overall task and are operating in the same environment, it is reasonable to assume some level of similarity in the distribution function of their modeled parameters. People vary in their behavior, so, obviously, there may be different types of owners: some will emphasize communication with their teams, and some will spend more time on map-based planning; some will dislike being disturbed while trying to evaluate their team's progress, while others may be more open to interruptions. Consequently, an owner's CA is likely to be able to find some CAs that are working with owners who are similar to its owner.

When adopting data collected by other agents, the two main questions are which agents the CA should rely on and to what extent it should rely on each of them. The selective-sharing mechanism relies on a statistical measure of similarity that allows the CA of any specific user to identify the similarity between its owner and other owners dynamically. Based on this similarity level, the CA decides if and to what degree to import other CAs' data in order to augment its direct observations, and thus to enable better modeling of its owner's characteristics.

It is notable that the cost of transferring observations between different CA modules of different agents is relatively small. This information can be transferred as part of regular negotiation communication between agents. The volume of such communication is negligible: it involves just the transmission of new observations' values.

In our learning mechanism, the CA constantly updates its estimation of the level of similarity between its owner and the owners represented by other CAs in the environment. Each new observation obtained either by that CA or any of the other CAs updates this estimation. The similarity level is determined using the Wilcoxon rank-sum test (Subsection 3.1).

Whenever it is necessary to produce a parameter estimate, the CA decides on the number of additional observations it intends to rely on for extracting its estimation. The number of additional observations to be taken from each other agent is a function of the number of observations it currently has from former interactions with its owner and the level of confidence the CA has in the similarity between its owner and other owners. In most cases, the number of observations the CA will want to take from another agent is smaller than the overall number of observations the other agent has; thus, it randomly samples (without repetitions) the required number of observations from this other agent's database. The additional observations the CA takes from other agents are used only to model its owner's characteristics. Future similarity level determination is not affected by this information augmentation procedure.

3.1 The Wilcoxon Test

We use a nonparametric method (i.e., one that makes no assumptions about the parametric form of the distributions each set is drawn from), because user characteristics in fast-paced domains do not have the structure needed for parametric approaches. Two additional advantages of a non-parametric approach are their usefulness for dealing with unexpected, outlying observations (possibly problematic for a parametric approach), and the fact that non-parametric approaches are computationally very simple and thus ideal for settings in which computational resources are scarce.

The Wilcoxon rank-sum test we use is a nonparametric alternative to the two-sample t-test [22, 14][2]. While the t-test compares means, the Wilcoxon test can be used to test the null hypothesis that two populations X and Y have the same continuous distribution. We assume that we have independent random samples $\{x_1, x_2, ..., x_m\}$ and $\{y_1, y_2, ..., y_n\}$, of sizes m and n respectively, from each population. We then merge the data and rank each measurement from lowest to highest. All sequences of ties are assigned an average rank. From the sum of the ranks of the smaller

[2]Chi-Square Goodness-of-Fit Test is for a single sample and thus not suitable.

sample, we calculate the test statistic and extract the level of confidence for rejecting the null hypothesis. This level of confidence becomes the measure for the level of similarity between the two owners. The Wilcoxon test does not require that the data originates from a normally distributed population or that the distribution is characterized by a finite set of parameters.

3.2 Determining Required Information

Correctly identifying the right number of additional observations to gather is a key determinant of success of the selective-sharing mechanism. Obviously, if the CA can identify another owner who has identical characteristics to the owner it represents, then it should use all of the observations collected by that owner's agent. However, cases of identical matches are likely to be very uncommon. Furthermore, even to establish that another user is identical to its own, the CA would need substantial sample sizes to have a relatively high level of confidence. Thus, usually the CA needs to decide how much to rely on another agent's data while estimating various levels of similarity with a changing level of confidence.

At the beginning of its process, the selective-sharing mechanism has almost no data to rely on, and thus no similarity measure can be used. In this case, the CA module relies heavily on other agents, in the expectation that all owners have some basic level of similarity in their distribution (see Section 2). As the number of its direct observations increases, the CA module refines the number of additional observations required. Again, there are two conflicting effects. On one hand, the more data the CA has, the better it can determine its level of confidence in the similarity ratings it has for other owners. On the other hand, assuming there is some difference among owners (even if not noticed yet), as the number of its direct observations increases, the owner's own data should gain weight in its analysis. Therefore, when CA_i decides how many additional observations, O_j^i should be adopted from CA_j's database, it calculates O_j^i as follows:

$$O_j^i = N * (1 - \alpha_{i,j})^{\sqrt{N}} + \frac{2 + ln(N)}{N} \quad (5)$$

where N is the number of observations CA_i already has (which is similar in magnitude to the number of observations CA_j has) and $\alpha_{i,j}$ is the confidence of rejecting the Wilcoxon null hypothesis.

The function in Equation 5 ensures that the number of additional observations to be taken from another CA module increases as the confidence in the similarity with the source for these additional observations increases. At the same time, it ensures that the level of dependency on external observations decreases as the number of direct observations increases. When calculating the parameter $\alpha_{i,j}$, we always perform the test over the interval relevant to the originating CA's distribution function. For example, when estimating the cost of interrupting the user, we apply the Wilcoxon test only for observations in the interval that starts from zero and ends slightly to the right of the formerly estimated RV (see Figure 1).

4. EMPIRICAL SETTING

We tested the selective-sharing mechanism in a system that simulates a distributed, Coordinators-like MAS. This testbed environment includes a variable number of agents, each corresponding to a single CA module. Each agent is assigned an external source (simulating an owner) which it periodically samples to obtain a value from the distribution being estimated. The simulation system enabled us to avoid unnecessary inter-agent scheduling and communication overhead (which are an inherent part of the Coordinators environment) and thus to better isolate the performance and effectiveness of the estimation and decision-making mechanisms.

The distribution functions used in the experiments (i.e., the distribution functions assigned to each user in the simulated environment) are multi-rectangular shaped. This type of function is ideal for representing empirical distribution functions. It is composed of k rectangles, where each rectangle i is defined over the interval (x_{i-1}, x_i), and represents a probability p_i, ($\sum_{i=1}^{k} p_i = 1$). For any value x in rectangle i, we can formulate $F(x)$ and $f(x)$ as:

$$f(x) = \frac{p_i}{x_i - x_{i-1}} \qquad F(x) = \sum_{j=1}^{i-1} p_j + \frac{(x - x_{i-1})p_i}{x_i - x_{i-1}} \quad (6)$$

For example, the multi-rectangular function in Figure 3 depicts a possible interruption cost distribution for a specific user. Each rectangle is associated with one of the user's typical activities, characterized by a set of typical interruption costs. (We assume the distribution of cost within each activity is uniform.) The rectangular area represents the probability of the user being engaged in this type of activity when she is randomly interrupted. Any overlap between the interruption costs of two or more activities results in a new rectangle for the overlapped interval. The user associated with the above distribution function spends most of her time in reporting (notice that this is the largest rectangle in terms of area), an activity associated with a relatively high cost of interruption. The user also spends a large portion of her time in planning (associated with a very high cost of interruption), monitoring his team (with a relatively small interruption cost) and receiving reports (mid-level cost of interruption). The user spends a relatively small portion of her time in scouting the enemy (associated with relatively high interruption cost) and resting.

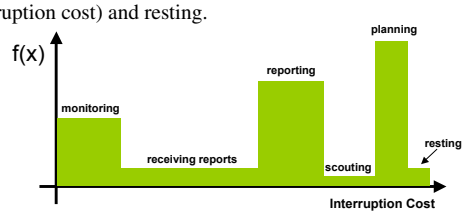

Figure 3: Representing interruption cost distribution using a multi-rectangular function

Multi-rectangular functions are modular and allow the representation of any distribution shape by controlling the number and dimensions of the rectangles used. Furthermore, these functions have computational advantages, mostly due to the ability to re-use many of their components when calculating the optimal reservation value in economical search models. They also fit well the parameters the CA is trying to estimate in fast-paced domains, because these parameters are mostly influenced by activities in which the user is engaged.

The testbed system enabled us to define either hand-crafted or automatically generated multi-rectangular distribution functions. At each step of a simulation, each of the CAs samples its owner (i.e., all CAs in the system collect data at a similar rate) and then estimates the parameter (either the expected cost when using the sequential interruption technique described in Section 2 or the probability that the owner will have the required information) using one of the following methods: (a) relying solely on direct observation ("self-learning") data; (b) relying on the combined data of all other agents ("average all"); and, (c) relying on its own data and selective portions of the other agents' data based on the selective-sharing mechanism described in Section 3.

5. RESULTS

We present the results in two parts: (1) using a specific sample environment for illustrating the basic behavior of the selective-sharing mechanism; and (2) using general environments that were automatically generated.

5.1 Sample Environment

To illustrate the gain obtained by using the selective-sharing mechanism, we used an environment of 10 agents, associated with 5 different interruptibility cost distribution function types. The table in Figure 4 details the division of the 10 agents into types, the dimensions of the rectangles that form the distribution functions, and the theoretical mean and reservation value (RV) (following Equation 3) with a cost $c = 2$ for sensing the interruption cost. Even though the means of the five types are relatively similar, the use of a reservation-value based interruption strategy yields relatively different expected interruption costs (RV, following Equation 2). The histogram in this figure depicts the number of observations obtained for each bin of size 1 out of a sample of 100000 observations taken from each type's distribution function.

Type	Agents	Rect.	Range	prob	mean	RV
I	1,2	1	0-20	0.40	50	14.1
		2	20-80	0.20		
		3	80-100	0.40		
II	3,4,5,6	1	0-40	0.25	50	25.3
		2	40-60	0.50		
		3	60-100	0.25		
III	7	1	0-80	0.10	85	56.6
		2	80-100	0.90		
IV	8,9	1	0-60	0.60	48	20.0
		2	60-90	0.40		
V	10	1	0-100	1.00	50	20.0

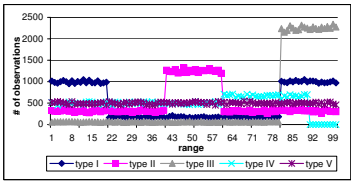

Figure 4: Users' interruptibility cost distribution functions (5 types)

Figure 5 gives CA performance in estimating the expected cost of interruption when using the reservation-value based interruption initiation technique. Each graph presents the average prediction accuracy (in terms of the absolute deviation from the theoretical value, so the lower the curve the better the performance) of a different type, based on 10000 simulation runs. The three curves in each graph represent the methods being compared (self-learning, average all, and selective-sharing). The data is given as a function of the accumulated number of observations collected. The sixth graph in the figure is the average for all types, weighted according to the number of agents of each type. Similarly, the following table summarizes the overall average performance in terms of the absolute deviation from the theoretical value of each of the different methods:

Iterations	Self-Learning	Averaging-All	Selective-Sharing	% Improvement[3]
5	20.08	8.70	9.51	53%
15	12.62	7.84	8.14	36%
40	8.16	7.42	6.35	22%

Table 1: Average absolute error along time

Several observations may be made from Figure 5. First, although the average-all method may produce relatively good results, it quickly reaches stagnation, while the other two methods exhibit continuous improvement as a function of the amount of accumulated data. For the Figure 4 environment, average-all is a good strategy for agents of type II, IV and V, because the theoretical reservation value of each of these types is close to the one obtained based on the aggregated distribution function (i.e., 21.27).[4] However, for types I and III for which the optimal RV differs from that value, the average-all method performs significantly worse. Overall, the sixth graph and the table above show that while in this specific environment the average-all method works well in the first interactions, it

[3]The improvement is measured in percentages relative to the self-learning method.

[4]The value is obtained by constructing the weighted aggregated distributed function according to the different agents' types and extracting the optimal RV using Equation 3.

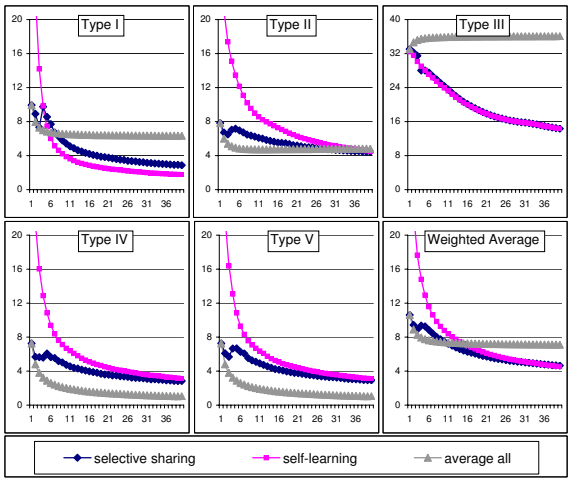

Figure 5: Average absolute deviation from the theoretical RV in each method (10000 runs)

is quickly outperformed by the selective-sharing mechanism. Furthermore, the more user observations the agents accumulate (i.e., as we extend the horizontal axis), the better the other two methods are in comparison to average-all. In the long run (and as shown in the following subsection for the general case), the average-all method exhibits the worst performance.

Second, the selective-sharing mechanism starts with a significant improvement in comparison to relying on the agent's own observations, and then this improvement gradually decreases until finally its performance curve coincides with the self-learning method's curve. The selective-sharing mechanism performs better or worse, depending on the type, because the Wilcoxon test cannot guarantee an exact identification of similarity; different combinations of distribution function can result in an inability to exactly identify the similar users for some of the specific types. For example, for type I agents, the selective-sharing mechanism actually performs worse than self-learning in the short term (in the long run the two methods' performance converge). Nevertheless, for the other types in our example, the selective-sharing mechanism is the most efficient one, and outperforms the other two methods overall.

Third, it is notable that for agents that have a unique type (e.g., agent III), the selective-sharing mechanism quickly converges towards relying on self-collected data. This behavior guarantees that even in scenarios in which users are completely different, the method exhibits a graceful initial degradation but manages, within a few time steps, to adopt the proper behavior of counting exclusively on self-generated data.

Last, despite the difference in their overall distribution function, agents of type IV and V exhibit similar performance because the relevant portion of their distribution functions (i.e., the "effective" parts that affect the RV calculation as explained in Figure 1) is identical. Thus, the selective-sharing mechanism enables the agent of type V, despite its unique distribution function, to adopt relevant information collected by agents of types IV which improves its estimation of the expected interruption cost.

5.2 General Evaluation

To evaluate selective-sharing, we ran a series of simulations in which the environment was randomly generated. These experiments focused on the CAs' estimations of the probability that the user would have the required information if interrupted. They used a multi-rectangular probability distribution function to represent

the amount of communication the user is engaged in with its environment. We models the growth of the probability the user has the required information as a function of the amount of communication using the logistic function,[5]

$$G(x) = \frac{1 + e^{\frac{-x}{12}}}{1 + 60e^{\frac{-x}{12}}}. \quad (7)$$

The expected (mean) value of the parameter representing the probability the user has the required information is thus

$$\mu = \int_{y=0}^{\infty} G(y)f(y)dy = \sum_{i=1}^{k} \left[\frac{x + 708ln(60 + e^{\frac{x}{12}})p_i}{60(x_i - x_{i-1})}\right]_{x_{i-1}}^{x_i} \quad (8)$$

where k is the number of rectangles used. We ran 10000 simulation runs. For each simulation, a new 20-agent environment was automatically generated by the system, and the agents were randomly divided into a random number of different types.[6] For each type, a random 3-rectangle distribution function was generated. Each simulation ran 40 time steps. At each time step each one of the agents accumulated one additional observation. Each CA calculated an estimate of the probability its user had the necessary information according to the three methods, and the absolute error (difference from the theoretical value calculated according to Equation 8) was recorded. The following table summarizes the average performance of the three mechanisms along different time horizons (measured at 5, 15 and 40 time steps):

Iterations	Self-Learning	Averaging-All	Selective-Sharing	% Improvement
5	0.176	0.099	0.103	41.4%
15	0.115	0.088	0.087	23.9%
40	0.075	0.082	0.065	13.6%

Table 2: Average absolute error along time steps

As can be seen in the table above, the proposed selective-sharing method outperforms the two other methods over any execution in which more than 15 observations are collected by each of the agents. As in the sample environment, the average-all method performs well in the initial few time steps, but does not exhibit further improvement. Thus, the more data collected, the greater the difference between this latter method and the two other methods. The average difference between selective-sharing and self-learning decreases as more data is collected.

Finally, we measured the effect of the number of types in the environment. For this purpose, we used the same self-generation method, but controlled the number of types generated for each run. The number of types is a good indication for the level of heterogeneity in the environment. For each number of types, we ran 10000 simulations. Figure 6 depicts the performance of the different methods (for a 40-observation collection period for each agent).

Since all simulation runs used for generating Figure 6 are based on the same seed, the performance of the self-learning mechanism is constant regardless of the number of types in the environment. As expected, the average-all mechanism performs best when all agents are of the same type; however its performance deteriorates as the number of types increases. Similarly, the selective-sharing mechanism exhibits good results when all agents are of the same type, and as the number of types increases, its performance deteriorates. However, the performance decrease is significantly more modest in comparison to the one experienced in the average-all mechanism.

[5]The specific coefficients used guarantee an S-like curve of growth, along the interval $(0, 100)$, where the initial stage of growth is approximately exponential, followed by asymptotically slowing growth.

[6]In this suggested environment-generation scheme there is no guarantee that every agent will have a potential similar agent to share information with. In those non-rare scenarios where the CA is the only one of its type, it will rapidly need to stop relying on others.

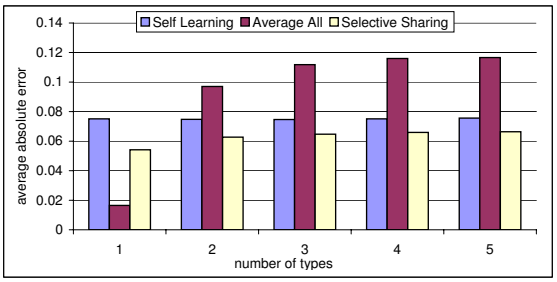

Figure 6: Average absolute deviation from actual value in 20 agent scenarios as a function of the agents' heterogeneity level

Overall, the selective-sharing mechanism outperforms both other methods for any number of types greater than one.

6. RELATED WORK

In addition to the interruption management literature reviewed in Section 2, several other areas of prior work are relevant to the selective-sharing mechanism described in this paper.

Collaborative filtering, which makes predictions (filtering) about the interests of a user [7], operates similarly to selective-sharing. However, collaborative filtering systems exhibit poor performance when there is not sufficient information about the users and when there is not sufficient information about a new user whose taste the system attempts to predict [7].

Selective-sharing relies on the ability to find similarity between specific parts of the probability distribution function associated with a characteristic of different users. This capability is closely related to clustering and classification, an area widely studied in machine learning. Given space considerations, our review of this area is restricted to some representative approaches for clustering. In spite of the richness of available clustering algorithms (such as the famous K-means clustering algorithm [11], hierarchical methods, Bayesian classifiers [6], and maximum entropy), various characteristics of fast-paced domains do not align well with the features of attributes-based clustering mechanisms, suggesting these mechanisms would not perform well in such domains. Of particular importance is that the CA needs to find similarity between functions, defined over a continuous interval, with no distinct pre-defined attributes. An additional difficulty is defining the distance measure.

Many clustering techniques have been used in data mining [2], with particular focus on incremental updates of the clustering, due to the very large size of the databases [3]. However the applicability of these to fast-paced domains is quite limited because they rely on a large set of existing data. Similarly, clustering algorithms designed for the task of class identification in spatial databases (e.g., relying on a density-based notion [4]) are not useful for our case, because our data has no spatial attributes.

The most relevant method for our purposes is the Kullback-Leibler relative entropy index that is used in probability theory and information theory [12]. This measure, which can also be applied on continuous random variables, relies on a natural distance measure from a "true" probability distribution (either observation-based or calculated) to an arbitrary probability distribution. However, the method will perform poorly in scenarios in which the functions alternate between different levels while keeping the "general" structure and moments. For example, consider the two functions $f(x) = (\lfloor x \rfloor mod 2)/100$ and $g(x) = (\lceil x \rceil mod 2)/100$ defined over the interval $(0, 200)$. While these two functions are associated with almost identical reservation values (for any sampling cost) and mean, the Kullback-Leibler method will assign a poor correlation between

them, while our Wilcoxon-based approach will give them the highest rank in terms of similarity.

While the Wilcoxon test is a widely used statistical procedure [22, 14], it is usually used for comparing two sets of single-variate data. To our knowledge, no attempt has been made yet to extend its properties as an infrastructure for determining with whom and to what extent information should be shared, as presented in this paper. Typical use of this non-parametric tool includes detection of rare events in time series (e.g., a hard drive failure prediction [17]) and bioinformatics applications (e.g., finding informative genes from microarray data). In these applications, it is used primarily as an identification tool and ranking criterion.

7. DISCUSSION AND CONCLUSIONS

The selective-sharing mechanism presented in this paper does not make any assumptions about the format of the data used or about the structure of the distribution function of the parameter to be estimated. It is computationally lightweight and very simple to execute. Selective-sharing allows an agent to benefit from other agents' observations in scenarios in which data sources of the same type are available. It also guarantees, as a fallback, performance equivalent to that of a self-learner when the information source is unique. Furthermore, selective-sharing does not require any prior knowledge about the types of information sources available in the environment or of the number of agents associated with each type.

The results of our simulations demonstrate the selective-sharing mechanism's effectiveness in improving the estimation produced for probabilistic parameters based on a limited set of observations. Furthermore, most of the improvement is achieved in initial interactions, which is of great importance for agents operating in fast-paced environments. Although we tested the selective-sharing mechanism in the context of the Coordinators project, it is applicable in any MAS environment having the characteristics of a fast-paced environment (e.g., rescue environments). Evidence for its general effectiveness is given in the general evaluation section, where environments were continuously randomly generated.

The Wilcoxon statistic used as described in this paper to provide a classifier for similarity between users provides high flexibility with low computational costs and is applicable for any characteristic being learned. Its use provides a good measure of similarity which an agent can use to decide how much external information to adopt for its assessments.

8. ACKNOWLEDGEMENT

The research reported in this paper was supported in part by contract number 55-000720, a subcontract to SRI International's DARPA Contract No. FA8750-05-C-0033. Any opinions, findings and conclusions, or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of DARPA or the U.S. Government. We are grateful to an anonymous AAMAS reviewer for an exceptionally comprehensive review of this paper.

9. REFERENCES

[1] P. Adamczyk, S. Iqbal, and B. Bailey. A method, system, and tools for intelligent interruption management. In *TAMODIA '05*, pages 123–126, New York, NY, USA, 2005. ACM Press.

[2] P. Berkhin. Survey of clustering data mining techniques. Technical report, Accrue Software, San Jose, CA, 2002.

[3] M. Ester, H. Kriegel, J. Sander, M. Wimmer, and X. Xu. Incremental clustering for mining in a data warehousing environment. In *Proc. 24th Int. Conf. Very Large Data Bases, VLDB*, pages 323–333, 24–27 1998.

[4] M. Ester, H. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *KDD-96*, pages 226–231, 1996.

[5] M. Fleming and R. Cohen. A decision procedure for autonomous agents to reason about interaction with humans. In *AAAI Spring Symp. on Interaction between Humans and Autonomous Systems over Extended Operation*, 2004.

[6] N. Friedman, D. Geiger, and M. Goldszmidt. Bayesian network classifiers. *Machine Learning*, 29:131–163, 1997.

[7] N. Good, J. Ben Schafer, J. Konstan, A. Borchers, B. Sarwar, J. Herlocker, and J. Riedl. Combining collaborative filtering with personal agents for better recommendations. In *AAAI/IAAI*, pages 439–446, 1999.

[8] K. Hinckley, J. Pierce, M. Sinclair, and E. Horvitz. Sensing techniques for mobile interaction. In *UIST '00*, pages 91–100, New York, NY, USA, 2000. ACM Press.

[9] E. Horvitz, C. Kadie, T. Paek, and D. Hovel. Models of attention in computing and communication: from principles to applications. *Commun. ACM*, 46(3):52–59, 2003.

[10] B. Hui and C. Boutilier. Who's asking for help?: a bayesian approach to intelligent assistance. In *IUI '06*, 2006.

[11] J. Jang, C. Sun, and E. Mizutani. *Neuro-Fuzzy and Soft Computing A Computational Approach to Learning and Machine Intelligence*. Prentice Hall, 1997.

[12] S. Kullback and R. Leibler. On information and sufficiency. *Ann. Math. Statist.*, 22:79–86, 1951.

[13] P. Maglio, T. Matlock, C. Campbell, S. Zhai, and B. Smith. Gaze and speech in attentive user interfaces. In *ICMI*, pages 1–7, 2000.

[14] H. Mann and D. Whitney. On a test of whether one of 2 random variables is stochastically larger than the other. *Annals of Mathematical Statistics*, 18:50–60, 1947.

[15] W. McClure. Technology and command: Implications for military operations in the twenty-first century. Maxwell Air Force Base, Center for Strategy and Technology, 2000.

[16] J. McMillan and M. Rothschild. Search. In Robert J. Aumann and Amsterdam Sergiu Hart, editors, *Handbook of Game Theory with Economic Applications*, pages 905–927. 1994.

[17] J. Murray, G. Hughes, and K. Kreutz-Delgado. Machine learning methods for predicting failures in hard drives: A multiple-instance application. *J. Mach. Learn. Res.*, 6:783–816, 2005.

[18] D. Sarne and B. J. Grosz. Estimating information value in collaborative multi-agent planning systems. In *AAMAS'07*, page (to appear), 2007.

[19] D. Sarne and B. J. Grosz. Timing interruptions for better human-computer coordinated planning. In *AAAI Spring Symp. on Distributed Plan and Schedule Management*, 2006.

[20] R. Vertegaal. The GAZE groupware system: Mediating joint attention in multiparty communication and collaboration. In *CHI*, pages 294–301, 1999.

[21] T. Wagner, J. Phelps, V. Guralnik, and R. VanRiper. An application view of coordinators: Coordination managers for first responders. In *AAAI*, pages 908–915, 2004.

[22] F Wilcoxon. Individual comparisons by ranking methods. *Biometrics*, 1:80–83, 1945.

[23] D. Zeng and K. Sycara. Bayesian learning in negotiation. In *AAAI Symposium on Adaptation, Co-evolution and Learning in Multiagent Systems*, pages 99–104, 1996.

[24] Y. Zhang, K. Biggers, L. He, S. Reddy, D. Sepulvado, J. Yen, and T. Ioerger. A distributed intelligent agent architecture for simulating aggregate-level behavior and interactions on the battlefield. In *SCI-2001*, pages 58–63, 2001.