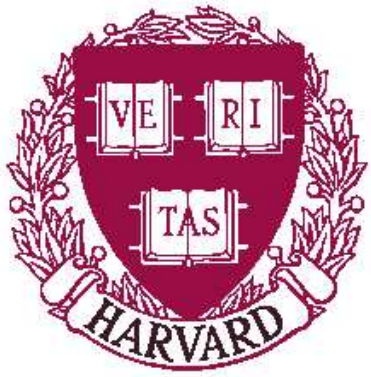
Sharing Experiences to Learn User Characteristics in Dynamic Environments with Sparse Data

David Sarne and Barbara Grosz

TR-15-06



Computer Science Group
Harvard University
Cambridge, Massachusetts

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

David Sarne and Barbara J. Grosz
Division 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 only made 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 with only a small number of observations 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" (CA) module of a coordination-manager system provided an empirical setting for evaluation. Simulation-based evaluations demonstrate 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.

1. INTRODUCTION

For many real-world scenarios, autonomous agents will need to operate in dynamic, uncertain environments having partial and incomplete information about the results of their actions and characteristics of other agents or humans 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 for use in situations in which this characteristic (or a property highly correlated with it) is 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 it 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, like disaster rescue, that are characterized by environments in which conditions may be rapidly changing, actions (whether of autonomous agents or of people) as well as 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 concurrently 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 environments in which agents must make decisions and take prompt action on an ongoing basis throughout the task or operation.

In fast-paced environments, information gathering may be limited, and there is not the possibility of learning offline nor of waiting to collect large amounts of data before making decisions. Thus fast-paced environments impose three constraints on any mechanism for learning a distribution function (including the large range of Bayesian update techniques [22]): (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 thus it attempts to minimize the overall error in its estimations. In such environments, an agent's individually acquired data (i.e., its own observations) is 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 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 distribution for two different agents is highly, positively correlated in terms of the values observed for a specific characteristic, 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 [16][2]), in which agents support a distributed scheduling

[1] Later sections explain the need to learn the distribution of the probabilistic parameter rather than just determining some value in the distribution. Learning the distribution is important whenever the overall shape of the distribution and not just individual features of it (such as mean) are important.

[2] The DARPA *Coordinators* research program aims to design, implement, and demonstrate a computational framework for distributed scheduling from which intelligent cognitive software agents can evolve that will help fielded military units adapt their mission plans and schedules as the situation around

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 they would be useful in any settings in which a group of agents undertake a task in a new environment and obtain observations of individual parameters each of them need for their decision-making at a similar rate.

In this paper we present a mechanism that was used for learning relevant key user's characteristics in environments of the types described above. 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 significantly improve the accuracy of its distribution-based estimations, in particular when the number of available observations is relatively small in comparison to relying only on the interactions with the specific user (subsequently denoted *"self-learning"*) or pooling all data unconditionally (subsequently, *"average all"*).

The mechanism was successfully tested using a system that simulates a *Coordinators*'s environment. The next section of the paper describes the problem of estimating user-related parameters in fast-paced domains, and in particular in the *Coordinators* environment, illustrating in more detail the need for the learning mechanisms developed in this paper. Section 3 provides an overview of the methods we developed. The implementation, empirical setting, and results are given in Section 4-5. A review of related work and a comparison with other methods (primarily from the area of clustering in machine learning) are given in Section 6. We conclude with a discussion in sections 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'" application domain [20]. In this domain, the 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 problems are distributed in nature and scheduling problems must be solved distributively. Each agent receives a different, typically only partial, local view of the different tasks and structures that constitute the full multi-agent problem. However, schedule revisions that affect more than one agent must be coordinated. Agents thus must share certain kinds of information, and in this 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.

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 distributed scheduling. As a result, the CA must estimate the expected benefit of any such interaction and the cost associated with it [17]. 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 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 for 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 such as the one described in [16].

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; 7, inter alia]. Research on interaction management has addressed the problem of timing interruptions appropriately using sensor-based statistical models of human interruptibility. Using sensed context, these systems attempt to infer the degree of distraction likely to be caused by an interruption. The main goal of such work is to reduce interruption "costs" by delaying interruptions to times that are "convenient". This work typically uses Bayesian models to learn the 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[7]. Other work focuses on such interruptibility-cost-influencing factors as user' frustration and distractability [8].

Prior work on interruption management deploys a sensor-based statistical models of human interruptibility. Using sensed context, the systems infer the degree of distraction likely to be caused by an interruption. These interruptibility estimates might then be used for balancing the estimated importance of the interaction against the degree of distraction likely to be caused by it. In addition to considering 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 (the complement of the situation we consider), this prior work differs from the fast-paced environments problem we address in two ways that fundamentally change the nature of the problem and hence alter the possible solutions. First, the interruptibility-estimation models are task-based. Second, they rely on continuous monitoring of a user's activities.

In fast-paced environments like disaster rescue, there is usually no single task structure, and it is difficult to determine the actual attentional state of agent-owners [13]. In such settings, owners must make complex decisions about current operations, which typically involve a number of other people in their units, while remaining reactive to events that diverge from expectations [23]. For instance, during disaster rescue, a first-response unit may begin rescuing survivors trapped in a burning house, when suddenly a wall collapses, forcing the unit to retract and re-plan their actions. Generally, there is not simply a single task being done, and some of the activities themselves may have little internal structure. Thus it does not suffice simply to do temporal reasoning about the best time to interrupt.

Prior work has tracked users focus of attention using a range of devices, including those able to monitor gestures [6] and track eye-gaze to identify focus of visual attention [11, 18], thus enabling estimations of cognitive load and physical indicators into 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 fast-paced environments will only allow for the activation of sensors for short periods of time on an ad-hoc basis, be-

them changes and impacts their plans [19].

cause agents' resources are severely limited and largely devoted to other tasks.

Methods that depend on predicting what a user 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 the user's attentional state being well structured (as a result of the complex of interleaved tasks which users are typically executing) and monitoring can only be done on a sporadic, non-continuous basis. Thus, at any given time, the cost of interaction with the user can be considered probabilistic, as reflected over a single random monitoring event, and can be assigned a probability distribution function. As a result, 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 it 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" [14]. It bases the CA's cost estimation on an integrated mechanism that uses 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 considers an individual interested in identifying an opportunity that will minimize its expected cost (or maximize its expected utility), while the search process itself is associated with a cost, and opportunities (e.g. interruption opportunities) are associated with a stationary distribution function. The search technique we use is sequential search strategy [14] in which the searcher draws one observation at a time, allowing multiple search stages. The dominating strategy in this model is a reservation-value based strategy. This strategy determines a lower bound, exploits any opportunity above the bound, but continues the search if it has only found a value below this bound. Since the interruption level is a random variable, that is associated with a stationary distribution function, knowing the value of an opportunity evaluated at a specific time does not affect the probability associated with any future opportunity value. Thus, MDPs are inapplicable in fast-paced environments.

Using the economic search framework, 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's CA module can activate sensing devices to get an estimation for the interruption cost, x, at the current time with a cost c (representing the cost of operating the sensing devices for a single time unit). The CA module sets a reservation value, which is the threshold for its strategy. As long as the sensor-based observation x is greater than this reservation value, it will wait a while and re-sample the user for a new estimation (again, with a cost c). The expected cost, $V(x_{rv})$, using such a strategy with a reservation value x_{rv} is described by the following equation:

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

Equation 1 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:

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

By substituting (2) in Equation 1 we obtain (after integration by parts):

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

from which the optimal reservation value, x^*_{rv}, and consequently (from Equation 2) also $V(x^*_{rv})$) can be computed.

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.) It is not always necessary to rely on complete similarity in the distribution function of different users. For some parameters, as in the case of the user's interruptibility level, it is enough to rely on similarity in the relevant portion of the distribution function for effectively sharing augmenting the observation database. The implementation (Sections 4-5) relies on this fact.

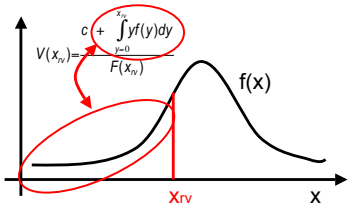


Figure 1: The distribution structure affecting the expected cost's calculation

2.2 Estimating the Probability of Having Information

One way to estimate the probability an owner will have information required by the agent (e.g., the user 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 owner, calculating the ratio between the number of times the owner had the information) and the total number of interactions. Alternatively, the CA can attempt to infer this probability from measurable characteristics of the owner's behavior, which it can assess without interrupting the owner. This alternative approach does not require any initial set of interactions with the user and is more useful in fast-paced domains. For example, the amount of interaction with their environment in which owners engage (e.g., their level of communication with others), which can be obtained without interrupting the owner, provides some indication of the frequency with which they obtain new information. Zhang et al. [23] describe a range of information that owners may obtain in *Coordinators*-like scenarios through communication channels. In addition to communication channels (formal and informal), other important tools for acquiring relevant external information include: occasional coordination meetings (e.g., used for reporting status of task execution), open communication the owner passively listens to (e.g. when leaving the radio open, one gets to listen to messages associated with other teams in the area) and direct communication used for coordination with other owners participating in the execution of the joint task (throughout which individual often informally learns about the status and existence of other actions being executed by others).

Given occasional updates it receives about the level of communication an owner maintains with its environment, the CA can estimate the probability that a random interaction with the owner will yield the information needed by the system. Thus, this alternative approach, which is the one used by the CA we designed, is an indirect one. The method uses owner-environment interactions as a kind of proxy for measuring whether the owner has certain information that could be gained through such interactions with the environment. The CA needs to estimate a particular user-property, in this case the probability of the owner having specific information. To do so, it learns the distribution function for the owner's level of involvement with the environment, which is less "costly" than repeated, direct interactions with the owner to probe whether or not the owner has the information. 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

In this section, we present the selective-sharing mechanism by which the CA learns the distribution function of a required probabilistic parameter by taking advantage of data collected by other CAs in its environment. The section first explains in more detail the need for increasing the number of observations used as the basis of estimation, and then explains the method used to determine how much data should be adopted 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 (i.e., self-learning). Based on this histogram, the CA can estimate the parameter either by taking into account the entire range of values (as in the case of estimating the mean) or a portion of it (as in the case of finding the expected cost when using a reservation-value-based strategy). However, 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 $c = 0.5$).

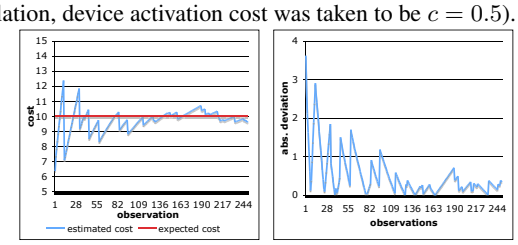


Figure 2: The convergence of a single CA to its optimal strategy

The 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 the CA reaches a very accurate estimation for the expected interruption cost. Nevertheless, 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. As argued above, the agent is severely limited in the number of times it can interact with its owner because of the nature of fast-paced domains. Therefore, 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.

One way to decrease the deviation from the actual value is by having the CA augment the data it acquires from observing its owner with observations made by other owners' agents for owners with similar distribution functions. 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 would hate 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 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 on-the-fly the similarity between its owner and other owners. Based on this similarity level, the CA can decide if and to what degree it wants to import other CA's data in order to augment its direct observations, 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 the "regular" negotiation communication between agents, and even if this will require designated communication then the volume of such communication is basically negligible: it involves just the transmission of new observations consisting of the observation value and the CA identification. The cost of such communication is definitely negligible in comparison to the misjudge in the generated estimation.

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 it 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 observation the other agent has, thus it randomly samples (without repetitions) the required number of observations from this other agent's database.

3.1 The Wilcoxon Test

We use a nonparametric method[3] (i.e. one that makes no assumptions about the parametric form of the distributions each set is drawn from), because the user-characteristics in fast-paced domains we are modeling 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 that might be problematic with a parametric approach, and the fact that non-parametric approaches are computationally very simple and thus ideal for settings in computational resources are scarce.

The Wilcoxon rank-sum test which we use is a nonparametric alternative to the two-sample t-test [21, 12]. 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 dis-

[3]Chi-Square Goodness-of-Fit Test is for a single sample thus not suitable for our case. Similarly, if we used the independent samples t-test the null hypothesis focuses on the population means while our goal is to check the null hypothesis that the distribution function also have equal shapes and equal dispersions.

tribution. 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 of each measurement from lowest to highest. All sequences of ties are assigned an average rank. From the sum of the ranks of the smaller sample we can calculate the test statistic and extract the level of confidence for rejecting the null hypothesis. This level of confidence becomes our measure for the level of similarity between the two owners. When using the Wilcoxon test we neither need to assume that the data originates from a normally distributed population, nor that the distribution is characterized by a finite set of parameters.

3.2 Determining Required Information

Getting the right decision on the number of additional observations to gather is a key element affecting the success of the selective-sharing mechanism. Obviously, if the CA can identify another owner who is identical in its characteristics to the owner it represents, then it should use all of the observation collected by the agent associated with this owner. 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 determine the similarity with a relative 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, because of the sampling), as the number of its direct observations increases, the owner's own data should gain weight in its analysis. Therefore, when the CA_i decides how many additional observations, O_j^i should be adopted from CA_j's database, it uses the following term:

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

where N is the number of observation CA_i already has (which is similar in its magnitude to the number of observation CA_j has) and $\alpha_{i,j}$ is the confidence of rejecting the Wilcoxon null hypothesis, concerning the distribution each of the two CAs attempts to model.

The function given in Equation 5 guarantees that the number of additional observations to be taken from other CA module instances 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 in external observations decreases as the number of direct observations increases. Notice that when calculating the parameter $\alpha_{i,j}$, we will 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 will apply the Wilcoxon test only for observation 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 *Coordinators* environment) 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) were 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), representing 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:

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

Consider, for example, the following multi-rectangular function in Figure 3, depicting the interruption cost distribution of a specific user:

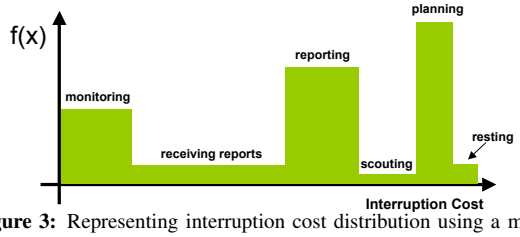


Figure 3: Representing interruption cost distribution using a multi-rectangular function

Each rectangle is associated with a typical activity the user is engaged in, characterized with a set of typical interruption cost values (notice that 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 it is randomly interrupted. Any overlap between the interruption cost 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 its 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 great openness to interruptions, reflected by the relatively small interruption cost) and receiving reports (variable cost of interruption). The user spends a relatively small portion of its time in scouting the enemy (associated with relatively high interruption cost) and resting (where no interruption is recommended whatsoever).

Multi-rectangular functions are modular and allows 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 its components when calculating the optimal reservation value in economical search models. The multi-rectangular function fits well the parameters the CA is trying to estimate in fast-paced domains since these are mostly influenced by the activities the user is engaged in.

The testbed system enabled us to define multi-rectangular distribution functions either hand-crafted or independently generated. 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 to be estimated (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 observations ("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

Before presenting the main experimental results, we provide a detailed analysis of a specific environment to illustrate the basic behavior of the selective-sharing mechanism described in this paper. The main experiments, in which the environment was automatically generated for each simulation run, indicate the expected performance of the system in general and a strong demonstration of its usefulness. To interpret these results, however, it is helpful to examine in more detail a particular case.

5.1 Detailed Illustration

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 to types and the structure of each type (the dimensions of the different rectangles that form the distribution function). The table also gives the theoretical mean and reservation value (RV) (calculated according to Equation 3)[4]. Notice that even though the mean of the 5 types is relatively similar, the use of a reservation-value based interruption strategy results with relatively different expected interruption costs (the expected interruption costs equals RV, according to 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

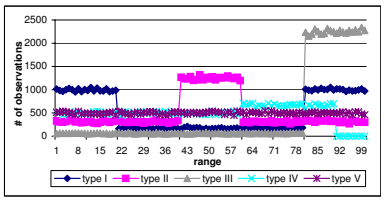


Figure 4: Users' interruptibility cost distribution functions (5 types)

Figure 5 depicts the CAs' 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, thus the lower the curve is the better the achieved performance) of a different type, based on 10000 simulation runs. The three curves in each graph represent the three different methods being compared (self-learning, average all, and selective sharing). The data is given as a function of the accumulated number of observation 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[5]
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. In the specific environment described in Figure 4, relying on all agents (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

[4]For this example we used $c = 2$ (cost of sensing the interruption cost).

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

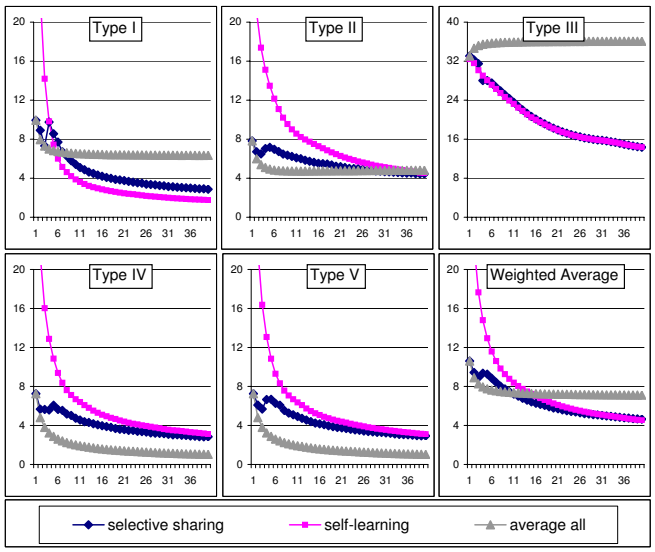


Figure 5: Average absolute deviation from the theoretical RV in each method (10000 runs)

to the one obtained based on the aggregated distribution function (i.e., 21.27).[6] On the other hand, for types I and III where the optimal RV is different from that value, the average-all method does significantly worse than the two other methods. Overall, the sixth graph and the table above show that while the average-all method in this specific environment is beneficial in the first interactions, it is being outperformed by the the selective-sharing mechanism of selective adoption of other agents' data. 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 the average all.

In the long run (and as shown in the following subsection for the general case), the average-all method exhibits the worst expected performance. 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 the 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. Lastly, notice that despite the difference in their overall distribution function, agents of type VI and V exhibit similar performance. This is 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

[6]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.

To extend our understanding of the performance of the selective-sharing mechanism in general, we ran further simulations, in which the environment was randomly generated. In these experiments, we focused on the CAs' attempt to estimate the probability that the user would have the required information if interrupted, and we used a multi-rectangular probability distribution functions to represent the amount of communication the user is engaged in with its environment. Then we applied the following logistic function, which models the growth of the probability the user has the required information as a function of the amount of communication[7]:

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

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 where on each simulation a 20-agents new environment was automatically generated by the system, and the agents were randomly divided to a random number of different types[8]. For each type, a random 3-rectangle distribution function was generated. Each simulation run 40 time steps, where on each time step each of the agents accumulated an additional observation. Over each time step each CA calculated its estimate for the probability the user has the necessary information according to the three methods, and the absolute error (in comparison to 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 as 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 any of the agents. Again, while the average-all method performs well over the first few time steps, it does not exhibit any improvement in further time steps, thus the more data collected the greater is the difference between this latter method and the two other methods. The average difference between the selective sharing and the self-learning method decreases as more data is collected, since the more data the agent has, the weaker is our tendency to adopt other agents' data for generating the estimation.

Finally, we attempted to measure the effect of the number of types in the environment over the different methods' performance. For this purpose, we used the same 20-agent environment with the self-generation functionality, but we 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 run 10000 simulations. Figure 6 depicts the performance of the different methods (again in terms of the absolute deviation of the estimate from the theoretical value, average over a 40-observation collection period for each agent) as a function of the different number of types used.

[7]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 a slowing growth, until growth stops.

[8]Notice that in the above 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.

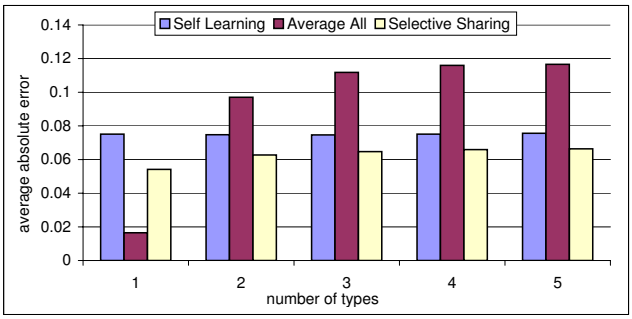


Figure 6: Average absolute deviation from actual value in 20 agents scenarios as a function of the agents' heterogeneity level

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. (In this method the agent relies only on its own collected observations thus is not affected by the other agents' types.) As expected, the average-all mechanism performs best when all agents are of the same type; however its performance immediately deteriorates and remains poor 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. Nevertheless, the performance decrease is significantly more modest in comparison to the one experienced in the average-all mechanism. Overall, the selective-sharing mechanism outperforms both other methods for any number of types greater than 1.

6. RELATED WORK

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

An important functionality on which the selective-sharing mechanism relies is the ability to find similarity between specific parts of the probability distribution function associated with a characteristic of different users. Such capability is closely related to clustering and classification, an area that is widely studied in machine learning. Given space considerations, our review of this area is by no means complete but emphasizes some representative approaches for clustering. In spite of the richness of available clustering algorithms (such as the famous K-means clustering algorithm [9], hierarchical methods, Bayesian classifiers [5], 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 predefined attributes. Additional difficulty in applying these distance-based mechanism is in defining the distance measure (most clustering techniques use for this measure the Euclidean distance formula).

Many clustering techniques were suggested in data mining (see survey [2]), and with particular focus on incremental updates of the clustering, due to the very large size of the databases [3]. However the applicability of these in our domain is quite limited due to their requirement to have a large set of existing data to rely on. Similarly, clustering algorithms designed for the task of class identification in spatial databases (e.g. relying on a density-based notion [4]) are non-applicable for our case since our data has no spatial attributes.

The most relevant method for our purposes is the Kullback-Leibler divergence (or relative entropy) index that is used in probability theory and information theory [10]. This measure, which can also be applied on continuous random variables, relies on a natural distance

measure from a "true" probability distribution (either observations-based or precisely calculated) to an arbitrary probability distribution. Nevertheless, the method will perform poorly in scenarios where 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 value (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 [21, 12] it is usually used for comparing two sets of single-variate data and to our knowledge no attempt has been made yet to extend its properties as an infrastructure for determining who 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 [15]) and many bioinformatics applications (e.g. finding informative genes from microarray data). Here, the statistical test is used primarily as an identification tool and ranking criterion for the affecting factors in the prediction model.

7. DISCUSSION AND CONCLUSIONS

The selective-sharing mechanism presented in this paper does not make any assumptions about the format of the data used, nor about the structure of the distribution function of the parameter to be estimated. It is computationally lightweight and very simple to execute. As can be seen from the detailed illustration that was given, the selective-sharing mechanism 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 exclusive in the environment. Furthermore, the selective-sharing mechanism does not require any prior knowledge about the type ("classes") of information sources available in the environment nor 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 the initial interactions, which is of great importance for agents operating in fast-paced environments. While we tested the selective-sharing mechanism for the CA module in the *Coordinators* project, it is applicable for any MAS environment having the characteristics of a fast-paced environment (e.g. first aiders in *Rescue* environments). 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, based on which an agent can decide how much external information each agent should adopt for its assessments.

Some specific activities or correlations between interruptibility levels can be modeled directly. For example, a CA module can estimate the time an owner will continue his current activity (e.g., the duration of radio communication). However, this possibility does not eliminate the need for indirect methods during the initial stages of an operation. Furthermore, the two approaches are complementary, not in conflict.

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

[6] 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.

[7] 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.

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

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

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

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

[12] 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.

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

[14] 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.

[15] 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.

[16] D. Sarne and B. Grosz. Estimating information value in collaborative multi-agent planning systems. Technical Report TR-16-06, Harvard University, 2006.

[17] 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.

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

[19] T. Wagner. The darpa/ipto coordinators program, 2005.

[20] 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.

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

[22] 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.

[23] 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.