Adjustable Autonomy in Real-world Multi-Agent Environments

Paul Scerri, David Pynadath, Milind Tambe
Information Sciences Institute and Computer Science Department
University of Southern California
4676 Admiralty Way, Marina del Rey, CA 90292
pausc@ida.liu.se, pynadath@isi.edu, tambe@isi.edu

ABSTRACT

Through *adjustable autonomy* (AA), an agent can dynamically vary the degree to which it acts autonomously, allowing it to exploit human abilities to improve its performance, but without becoming overly dependent and intrusive in its human interaction. AA research is critical for successful deployment of multi-agent systems in support of important human activities. While most previous AA work has focused on individual agent-human interactions, this paper focuses on *teams* of agents operating in real-world human organizations. The need for agent teamwork and coordination in such environments introduces novel AA challenges. First, agents must be more judicious in asking for human intervention, because, although human input can prevent erroneous actions that have high team costs, one agent's inaction while waiting for a human response can lead to potential miscoordination with the other agents in the team. Second, despite appropriate local decisions by individual agents, the overall team of agents can potentially make global decisions that are unacceptable to the human team. Third, the diversity in real-world human organizations requires that agents gradually learn individualized models of the human members, while still making reasonable decisions even before sufficient data are available. We address these challenges using a multi-agent AA framework based on an adaptive model of users (and teams) that reasons about the uncertainty, costs, and constraints of decisions at *all* levels of the team hierarchy, from the individual users to the overall human organization. We have implemented this framework through Markov decision processes, which are well suited to reason about the costs and uncertainty of individual and team actions. Our approach to AA has proven essential to the success of our deployed multi-agent Electric Elves system that assists our research group in rescheduling meetings, choosing presenters, tracking people's locations, and ordering meals.

1. INTRODUCTION

In many recent exciting, ambitious applications of agent technologies, individual agents or teams of agents act in support of the critical activities of individual humans or even entire human organizations. Application areas range from intelligent homes [8], to "routine" organizational coordination [11], to electronic commerce [1], to long-term space missions [2]. These new applications have fostered an increasing interest in *adjustable autonomy* (AA), i.e., in agents that *dynamically adjust their own level of autonomy based on the situation* [4]. Essentially, an agent can give up its autonomy by transferring decision-making control to a human for improved task performance. The key issue is to transfer control intelligently, harnessing human skills, knowledge, preferences, etc. as appropriate, without overly burdening the humans.

When agents are embedded in large human organizations, not only do agents interact with humans, they also coordinate with each other and act jointly in teams. However, existing AA techniques focus on the interactions between only an individual agent and a single human user. Thus, the requirements of teamwork and coordination give rise to novel AA challenges not addressed by previous research. We focus in particular on three key novel challenges: the *AA coordination challenge*, the *AA team decision challenge*, and the *AA safe learning challenge*.

The *AA coordination challenge* arises during the transfer of decision-making control. Determining when an agent should transfer control to a human (or vice versa) is a central, well-known AA problem. The novel challenge for team settings is for agents to avoid miscoordination with teammates during such a transfer, while simultaneously minimizing the risk of costly errors. Techniques from previous AA research fail to address this challenge. For instance, one technique uses uncertainty as the lone rationale for transferring decision-making control, relinquishing control to humans whenever uncertainty is high [5]. In a team setting, the agent cannot transfer control so freely, because as the agent waits for a human response, its teammates expect it to still fulfill its responsibilities to the overall joint task. As an example, consider an agent that manages an individual user's calendar and can request the rescheduling of a team meeting if it thinks its user will be unable to attend on time. Rescheduling is costly, because it disrupts the calendars of the other team members, so the agent can ask the user for confirmation to avoid making an unnecessary rescheduling request. However, while it waits for a response, the other users will begin arriving at the meeting room, and if the user does not arrive, they will waste their time waiting as the agent sits idly by, doing nothing. On the other hand, if, despite the uncertainty, the agent acts autonomously and informs the others that its users cannot attend, then its decision may still turn out to be a grave mistake. Thus, the *AA coordination challenge* requires that an agent weigh possible team miscoordination while waiting for a human response against possible erroneous actions as a result of uninformed decisions.

AGENTS'01, May 28-June 1, 2001, Montréal, Quebec, Canada.

The *AA team decision challenge* arises due to the multiple levels of decision-making in teams, where subteams make decisions based on individual agent input, teams make decisions based on subteam input, etc. Unfortunately, despite responsible decisions by individual team members, the agent team's collective decision may be highly undesirable to the human team. The agents thus face the problem of making a teamwide decision, even though the human users interact with only individual agents. Therefore, the AA framework must address decisions made at all levels of a team hierarchy.

The increased need for autonomous action in teams may force agents to act despite insufficient experience, potentially causing team failures when they are incorrect. While noisy data are a common learning issue, the AA coordination challenge prevents us from completely relying on the escape hatch from previous research — namely, asking human users when unsure [9]. For instance, in our initial implementation of agents to monitor meetings (described in Section 3), one user's agent learned rules that led it to autonomously cancel an important meeting. The *AA safe learning challenge* is to allow agents to learn more accurate models of their users, while still protecting users from temporary anomalies during the course of that learning.

We have conducted our research on AA in the context of a real-world multi-agent system, called *Electric Elves* (E-Elves) [11], that we have used for several months at USC/ISI. E-Elves assists a group of 9 researchers and one project assistant in their daily activities, providing a unique, exciting opportunity to test ideas in a real environment.

To address the *AA coordination challenge*, E-Elves agents explicitly reason about team coordination via a novel three-step approach. First, before transferring decision-making control, an agent explicitly weighs the cost of waiting for user input and any concomitant potential team miscoordination against the cost of erroneous autonomous action. Second, agents do not rigidly commit to transfer-of-control decisions (as is commonly done in previous work), but instead reevaluate decisions as required for continued team coordination (e.g., if an agent transfers control to a user, but the user fails to act, the agent can act to avoid team miscoordination). Third, when an agent takes control of a decision to avoid miscoordination it may be faced with significant costs and uncertainty. Rather than force a risky decision, an agent can elect to change coordination arrangements, postponing or reordering activities, to "buy time" to lower decision cost/uncertainty. Since a sequence of coordination changes may be needed, and since each coordination change is costly, agents look ahead at possible sequences of coordination changes, selecting one that maximizes team benefits.

Two approaches are possible in tackling the *AA team decision challenge*. One approach is to explicitly consider the team-level decision within the decision-making done by each individual agent. This approach can ensure the desired (sub)team decisions, but it greatly complicates the individual agents' AA reasoning. We take a different approach in E-Elves by simplifying the individual agent reasoning, but also introducing AA at the team level, allowing the agent team to consult a user team if the collective decision involves significant uncertainty, cost, etc. A key novelty at the team level is that the AA reasoning focuses on collective team features (e.g., *majority of team members*) rather than specific individual features.

We address the *safe learning challenge* in two ways: (i) We provide the agents with a significant amount of domain knowledge, providing a solid base on which to build; (ii) We augment the learning algorithm with a *safety net* that ensures that, even in the presence of noisy data, agents cannot learn harmful behaviors (which we explicitly identify beforehand).

Our approach to AA in E-Elves uses Markov decision processes (MDPs) [10]. MDPs allow explicit representation of individual and team decision costs and explicit representation and reasoning about uncertainty of the world state and user intentions. MDPs can also look ahead to allow agents to find optimal sequences of coordination changes.

2. ELECTRIC ELVES

Software agents now proliferate in human organizations, helping with information gathering, activity scheduling, email management, etc. The Electric Elves effort at USC/ISI is taking the next step: dynamic teaming of heterogeneous agents, including human proxies, to support the functioning of entire organizations[1].

As a step towards this goal, USC/ISI have had an agent team of 15 agents, including 10 proxies (for 10 people), running 24/7 for the past four months. We sketch the overall design of the system in Figure 1a. Each proxy is called Friday (from Robinson Crusoe's servant Friday) and acts on behalf of its user in the agent team. If a user is delayed to a meeting, Friday can reschedule the meeting, informing other Fridays, who in turn inform their human users. If there is a research presentation slot open, Friday may respond to the invitation to present on behalf of its user. Friday can also order its user's meals (see Figure 1b) and track the user's location, posting it on a Web page. Friday communicates with users using wireless devices, such as personal digital assistants (PALM VIIs) and WAP-enabled mobile phones, and via user workstations. Figure 1c shows a PALM VII connected to a Global Positioning Service (GPS) device, for tracking users' locations and enabling wireless communication with Friday.

Each Friday's team behavior is based on a teamwork model, called STEAM [14]. The Fridays model each meeting as a team's joint intention that, by the rules of STEAM, they keep each other informed about (e.g., a meeting is delayed, cancelled, etc.). Furthermore, Fridays use STEAM role relationships to model the relationships among team members. For instance, the presenter role is critical since the other attendees depend on someone giving a presentation. Thus, if the presenter cannot attend, the team recognizes a critical role failure that requires remedial attention.

The basic design of the Friday proxies is discussed in detail elsewhere [15] (where they are referred to as TEAMCORE proxies), but there have been several significant advances since that initial report. Previously, dynamic role allocation was on a first-come-first-served basis [14]. Now, the team auctions off the role, allowing it to consider complex combinations of factors and assign the best-suited agent. Figure 2a shows the auction tool that allows human users to view auctions in progress and intervene if they so desire. For example, for the role of presenter at a particular meeting, Friday bids on behalf of its user, indicating whether its user is capable or willing to present on the given presentation topic. In the auction in progress in Figure 2a, Jay Modi's Friday has bid that Jay is capable of giving the presentation, but is unwilling to do so. Friday looks up its user's (binary) capability to perform the role and submits a capability bid. While the capability bidding is fully autonomous, the willingness bidding is not (see Section 4.3). In the auction shown, Paul Scerri's agent has the highest bid and was declared the winner.

AA is of critical importance in Friday agents. Clearly, the more autonomous Friday is, the more time it saves its user. However,

[1]The Electric Elves project is a large collaborative effort at USC/ISI involving several research groups[11]; this paper focuses on the research that our group has conducted in adjustable autonomy in the context of this project

(a) Overall proxy-based architecture (b) Friday ordering dinner from CPK (c) Palm VII and GPS

Figure 1: Some of the devices and tools used in Electric Elves.

Friday has the potential to make costly mistakes when acting autonomously (e.g., volunteering an unwilling user for a presentation). Thus, each Friday must make intelligent decisions about when to consult its user and when to act autonomously. Furthermore, Friday faces significant, unavoidable uncertainty (e.g., if a user is not at the meeting location at meeting time, does s/he plan to attend?).

In addition to uncertainty and cost, the E-Elves raises the three novel AA challenges mentioned in Section 1. First, consider the *AA coordination challenge*. Suppose that, when faced with uncertainty, a Friday agent consults its user (e.g., to check whether the user plans to attend a meeting), but the user, caught in traffic, fails to respond. While waiting for a response, Friday may miscoordinate with its teammates (other Friday agents), since it fails to inform them whether the user will attend the meeting. This, in turn means that other meeting attendees (humans) waste their time waiting. Conversely, if, to maintain coordination, Friday tells the other Fridays that its user will not attend the meeting, but the user does indeed plan to attend, the human team suffers a potentially serious cost from receiving this incorrect information. Friday must instead make a decision that makes the best tradeoff possible between the possible costs of inaction and the possible costs of incorrect action.

The *AA team decision challenge* is to ensure effective team-level decision-making. While Friday's AA may ensure effective individual decisions, consequent agent team decisions may still be undesirable to the human team. As a simple example, suppose all Fridays act responsibly, and, as a result, the team has received a reasonable bid for an unallocated role, but outstanding bids still remain (possibly due to slow user response). The team faces a difficult choice: should it immediately assign the role, or should it wait in case it eventually receives a better bid? Delaying the decision gives the assignee less time to prepare for the presentation, but assigning someone immediately may mean a suboptimal assignment if a more suited user would have bid in the future.

Finally, the *safe learning challenge* requires protecting against temporary learning aberrations. The next section provides a list of failures that clearly demonstrates that faulty learning is indeed a significant issue in E-Elves.

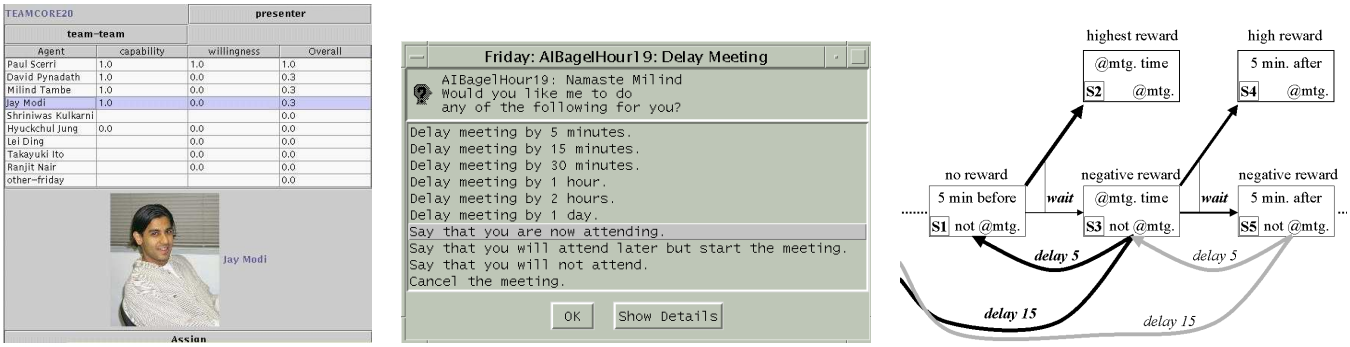
3. DECISION-TREE APPROACH

Our first attempt at AA in E-Elves was inspired by CAP [9], an agent system for helping a user schedule meetings. Like CAP, Friday learned user preferences using C4.5 decision-tree learning [13]. In training mode, Friday recorded values of a dozen carefully selected attributes and the user's preferred action (identified by query via a dialog box, as in Figure 2b) whenever it had to make a decision. Friday used the data to learn a decision tree (e.g., *if* the user has a meeting with his or her advisor, but is not at ISI at the meeting time, *then* delay the meeting 15 minutes). Also in training mode, Friday asked if the user wanted such decisions taken autonomously in the future. Friday again used C4.5 to learn a second decision tree from these responses. The key idea was to resolve the transfer-of-control decision by learning from user input.

Initial tests with the above setup were promising [15], but a key problem soon became apparent. When Friday encountered a decision it should not take autonomously, it would wait indefinitely for the user to make the decision, even though this inaction led it to miscoordinate with its teammates. To address this problem, if a user did not respond within a fixed time limit, Friday took an autonomous action. Although results improved, when we deployed the resulting system 24/7, it led to some dramatic failures, including:

1. Tambe's (a user) Friday incorrectly, autonomously cancelled a meeting with the division director. C4.5 over-generalized from training examples.
2. Pynadath's (another user) Friday incorrectly cancelled the group's weekly research meeting. A time-out forced the choice of an (incorrect) autonomous action when Pynadath did not respond.
3. A Friday delayed a meeting almost 50 times, each time by 5 minutes. The agent was correctly applying a learned rule but ignoring the nuisance to the rest of the meeting participants.
4. Tambe's proxy automatically volunteered him for a presentation, though he was actually unwilling. Again, C4.5 had over-generalized from a few examples and when a timeout occurred had taken an undesirable autonomous action.

From the growing list of failures, it became clear that the approach faced some fundamental problems. The first problem was the *AA coordination challenge*, which requires the agent balance potential team miscoordination against the cost of erroneous actions. Learning from user input, when combined with timeouts, failed to address the challenge, since the agent sometimes had to take autonomous actions although it was ill-prepared to do so (examples 2 and 4). Second, the approach did not consider the team cost of erroneous autonomous actions (examples 1 and 2). Effective agent AA needs explicit reasoning and careful tradeoffs when dealing with the different individual and team costs and uncertainties. Third, decision-tree learning lacked the lookahead ability to plan actions that may work better over the longer term. For instance, in example 3, each five-minute delay is appropriate *in isolation*, but the rules did not consider the ramifications of one action on successive actions. Planning could have resulted in a one-hour delay instead of many five-minute delays. Planning and consideration of cost could also lead to an agent taking the low-cost action of a short meeting delay while it consults the user regarding the higher-cost cancel action (example 1). Interestingly, this original effort did not



(a) Electric Elves auction tool (b) Dialog for delaying meetings (c) A small portion of simplified version of the delay MDP, with no response from the user

Figure 2: Auction tool, dialog box, and delay MDP used in E-Elves.

address the *team decision challenge*, given that it did not overcome the *AA coordination challenge*.

Potentially, the problems with the C4.5 approach could have been overcome with more data, a better set of attributes, and reasoning about uncertainty using C4.5 confidence factors. Unfortunately, this very rich domain makes it difficult to gather sufficient training data, so much of the required domain knowledge should be encoded a priori. The point is not that learning is inapplicable in AA (see Section 4.4), rather that, in a team context, the AA problem has significant complexities that preclude simple solutions.

4. ADDRESSING CHALLENGES IN AA

MDPs were a natural choice for addressing the issues identified in the previous section: reasoning about the costs of actions, handling uncertainty, planning for future outcomes, and encoding domain knowledge. To overcome computational complexity costs, we rely on partitioning the MDPs. Thus, a Friday has a separate MDP for each decision that it makes to reschedule meetings or volunteer a user for presentation or order meals. Overall, there are four types of MDPs. Section 4.1 discusses the basic representation of the MDPs, using the *delay MDP* as an example (two other MDPs, the *bid-for-role MDP* and the *order-meal MDP*, are similar). Section 4.2 illustrates how these MDPs address the *AA coordination challenge*. Section 4.3 discusses the *assign-role MDP*, which focuses on the *AA team decision challenge*. Finally, Section 4.4 discusses how our framework addresses the *safe learning challenge*.

4.1 MDP Representation

The delay MDP, typical of MDPs in Friday, represents a class of MDPs covering all types of meetings for which the agent may take rescheduling actions. For each meeting, an agent can autonomously perform any of the 10 actions shown in the dialog of Figure 2b (e.g., it could ask for a meeting delay of 15 minutes, announce the user will not attend, etc.). It can also wait, i.e., sit idly without doing anything, or can reduce its autonomy and ask its user for input (the user can choose from any of the 10 responses from Figure 2b).

The delay MDP reasoning is based on a world state representation, the most salient features of which are the user's location and the time. Figure 2c shows a portion of the state space, showing only the location and time features, as well as some of the state transitions (a transition labeled "delay n" corresponds to the action "delay by n minutes"). Each state also has a feature representing the number of previous times the meeting has been delayed and a feature capturing what the agent has told the other Fridays about the user's attendance (either nothing, that the user is currently attending, that the user will be late, or that the user will not be attending). There are a total of 768 possible states per each individual meeting.

The delay MDP's reward function has a maximum in the state where the user is at the meeting location when the meeting starts. A component of the reward, denoted r_{user}, focuses on the user attending the meeting at the meeting time, giving the agent incentive to delay meetings when its user's late arrival is possible. However, in isolation, r_{user} could drive the agent to choose arbitrarily large delays, virtually ensuring the user is at the meeting when it starts, but forcing other attendees to rearrange their schedules – a very costly activity. The team cost is considered by incorporating a negative reward, denoted r_{repair}, with magnitude proportional to the number of delays so far and the number of attendees, into the delay reward function. A pre-learning constraint (see Section 4.4) allows no more than 4 delays to a meeting, regardless of what transition probabilities are learned.

However, explicitly delaying a meeting may benefit the team, since without a delay, the other attendees may waste time waiting for the agent's user to arrive. Therefore, the delay MDP's reward function includes a component, r_{time}, that is negative in states after the start of the meeting if the user is absent, but positive otherwise. The magnitude of this reward, like r_{repair}, is proportional to the number of attendees. The reward function also includes a component r_{role}, which, like r_{user}, is positive in states where the user is in attendance and zero otherwise. However, the magnitude of r_{role} also increases with the importance of the user's role (e.g., speaker vs. passive participant) to the success of the meeting, thus representing the value of the user's attendance to the *team*. Finally, the reward function includes a component, $r_{meeting}$, which is positive once the meeting starts and zero everywhere else to deter meeting cancellation. The overall reward function for a state *s* is a weighted sum of the components:

$$r(s) = \lambda_{user} r_{user}(s) + \lambda_{repair} r_{repair}(s) + \lambda_{time} r_{time}(s) + \lambda_{role} r_{role}(s) + \lambda_{meeting} r_{meeting}(s)$$

Although taking into account team costs, Friday's decisions are on behalf of its user only; the team may not concur. Thus, even if Friday reasons about individual and team costs when requesting a delay, the team, after due deliberation, may not accept such a request, as discussed in Section 4.3.

The delay MDP's state transitions are associated with the probability that a given user movement (e.g., from office to meeting location) will occur in a given time interval. Figure 2c shows multiple transitions due to a 'wait' action, with the relative thickness of the arrows reflecting their relative probability. The MDP designer

encodes the initial probabilities, which our learning algorithm (described in Section 4.4) then customizes. In practice, transitions to states where the user arrives on time are highly likely.

The "ask" action, through which the agent gives up autonomy and queries the user, has two possible outcomes. First, the user may not respond at all, in which case, the agent is performing the equivalent of a "wait" action. Second, the user may respond, with one of the 10 responses from Figure 2b. A communication model [15] provides the probability of receiving a user's response in a given time step. The cost of the "ask" action is derived from the cost of interrupting the user. The probabilities and costs vary with the mode of interaction (e.g., a dialog box on the user's workstation is cheaper than sending a page to the user's cellular phone). We compute the expected value of user input by summing over the value of each possible response, weighted by its likelihood (computed using the delay MDP as a model of the user's decision-making). The value of each user response is computed using the reward function mentioned above, but assuming the user response is accurate. For instance, if the user provides input suggesting a 5-minute delay, then the agent knows that it will incur the cost of the 5-minute delay but will then receive the maximum reward when the user arrives at the (rescheduled) meeting on time.

Given the state space, actions, transition probabilities, and reward function of the MDP, a standard value iteration technique is used to compute an optimal policy, $policy^*(s)$:

$$policy^*(s) = \arg\max_a \Sigma_j M^a_{sj} U(j)$$

where $U(s)$, the utility of being in state s, is :

$$U(s) = r(s) + \max_a \Sigma_j M^a_{sj} U(j)$$

and $M^a_{sj} U(j)$ is the transition probability between state s and j given that action a is taken [12].

One possible policy, generated for a subclass of possible meetings, specifies "ask" and then "wait" in state **S1** of Figure 2c, i.e., the agent gives up some autonomy. If the world reaches state **S3**, the policy again specifies "wait", so the agent continues acting without autonomy. However, if the agent then reaches state **S5**, the policy chooses "delay 15", which the agent then executes autonomously. However, the exact policy generated by the MDP will depend on the exact probabilities and costs used (see Section 5).

4.2 The AA coordination challenge

The delay MDP enables Friday to address the *AA coordination challenge* using the three-step approach introduced in Section 1. The previous section describes how the agent achieve the first step of balancing individual and team rewards, costs, etc. This section describes how such MDPs support the other two steps of our AA coordination approach as well.

The second step of our approach requires that agents avoid rigidly committing to transfer-of-control decisions. For example, if the agent decides to give up autonomy for a decision, it should not wait indefinitely for a user response, as a slow response could jeopardize the team activity. Instead, the agent must continuously reassess the developing situation, possibly changing its previous autonomy decisions (as specified in state **S5** by the policy discussed in the previous section). The MDP representation supports this by generating an autonomy *policy* rather than an autonomy *decision*. The policy specifies optimal actions for each state, so the agent can respond to any state changes by following the policy's specified action for the new state. In this respect, the agent's AA is an ongoing process, as the agent acts according to a policy throughout the entire sequence of states it finds itself in.

The third step of our approach arises because an agent may need to act autonomously to avoid miscoordination, yet it may face significant uncertainty and risk when doing so. In such cases, an agent can carefully plan a change in coordination (e.g., delaying actions in the meeting scenario) by looking ahead at the future costs of team miscoordination and those of erroneous actions. The delay MDP is especially suitable for producing such a plan because it generates policies after looking ahead at the potential outcomes. For instance, the delay MDP supports reasoning that a short delay "buys time" for a user to respond to a query from an agent, potentially reducing the uncertainty surrounding a costly decision, albeit at a small cost. Thus, an agent might choose a 15-minute delay to give time for an absent user to arrive, or respond, before cancelling a meeting.

Furthermore, the lookahead in MDPs enables effective long-term solutions to be found. As already mentioned the cost of rescheduling, r_{repair}, increases as more such repair actions occur. This provides a compact scheme for supporting some history dependency in the cost of future states. Thus, even if the user is very likely to arrive at the meeting in the next 5 minutes, the uncertainty associated with that particular state transition may be sufficient, when coupled with the cost of subsequent delays if the user does not arrive, for the delay MDP policy to specify an initial 15-minute delay (rather than risk three 5-minute delays). Thus, the agent reasons about the likelihood and cost of possible subsequent delays.

4.3 The AA Team Decision Challenge

Once individual team members provide their input to the team (e.g., suggestions to delay meetings or bids for roles), the team makes a collective decision based on that input. As discussed in Section 1, the *AA team decision challenge* refers to ensuring the effectiveness of such team decisions. In particular, despite individual Friday's responsible actions, the team may reach a highly undesirable decision. One approach suggested in previous work is to improve the decision-making at the individual level so as to avoid such team-level problems [6]. Unfortunately, this makes the individual reasoning more complex, because an agent would need to model and predict the actions of its teammates. We instead propose the novel approach to introduce AA at all levels of decision-making in a team hierarchy, i.e., at individual, subteam, and team level, enabling the agent team to tap into the human team's expertise and knowledge in difficult decisions. Once again, while consulting human team, the agent team must not overburden the users.

In a team context, the AA problem is for the team to decide whether to rely on its own reasoning or to relinquish the control to the human team. Clearly, in some cases, STEAM's teamwork rules, which form the basis of the team reasoning, have some (low) likelihood of error, perhaps due to inadequate input. Thus, consulting with a human team may be appropriate in some situations. For instance, a STEAM decision based on given role relationships may be accurate, but there may be errors in the original modeling of the role relationships themselves. Furthermore, each such team decision error has different cost.

Thus, analogous to the AA at the individual level, the team-based AA needs to reason about uncertainty, cost, and potential developments in the world. Hence, we again chose MDPs for the AA reasoning mechanism. Like the individual MDPs described in Section 4.2, the team MDPs compare the expected value of consulting the human team against the expected value of making an autonomous decision.

The team-level decision to delay a meeting is of sufficiently low uncertainty and low cost that it is always taken autonomously. In contrast, the team-level decision to close an auction and assign a presenter for a talk has high uncertainty and cost, so the agent team

will sometimes need to consult with the human team. The team can take one of two actions, closing the auction and assigning the role or waiting. The "wait" action also allows a human to make an assignment, so it is synonymous with asking the human team. The states of the team-level MDP have abstract team features rather than individual features (e.g., *few*, *half*, *many*, or *most* bids have been submitted for a role). In order to reason about the course of action that will maximize reward, the agent team needs to know the probability distribution for higher quality bids arriving in the subsequent time steps. This distribution is encoded a priori and then adjusted autonomously from observations. In each auction state s, the team receives reward:

$$r(s) = \lambda_{bid} r_{bid}(s) + \lambda_{accept} r_{accept}(s) + \lambda_{time} r_{time}(s)$$

that has a maximum when the team assigns a clearly superior, high-quality bidder to the role at the optimal time. The reward r_{bid}, proportional to the quality of the winning bid, encourages the team to leave the auction open until a high-quality bid comes in. The reward r_{accept} is positive if there is a clearly best bid and negative if there is no clearly best bid, discouraging the agent team from making assignments when there is uncertainty surrounding which bid is best. The reward r_{time} is based on how appropriate the timing of the assignment is — too early and the team may miss out on receiving better bids, too late and the assigned user will not have sufficient time to prepare. The r_{time} reward captures any time pressure associated with the role, encouraging the team to make an assignment earlier (e.g., to give the assigned presenter more time to prepare).

4.4 The AA Safe Learning Challenge

An accurate model of the user is critical to good AA reasoning. We create an initial "generic" user model, encoding our domain knowledge in an initial MDP that allows the agents to function well straight "out of the box". Improvements, via learning, to the MDP model lead to improved AA reasoning, primarily by personalizing the reasoning to particular users. Fridays learn via a simple reinforcement learning algorithm that periodically examines the extensive logs created during system execution and then updates parameters in the MDP user model accordingly. However, in a deployed system, anything learned has an immediate effect on the users — a phenomenon we have seen to be sometimes harmful in practice, as described in Section 3. Hence, we require a safety mechanism that protects the user from unacceptable policies learned due to noise, inadequate sensing, or limited training examples. We take a two-pronged approach to the *AA safe learning challenge*: (i) building substantial knowledge into the agents' generic user models (as described in Sections 4.1–4.3); and (ii) providing a safety net to prevent harmful policies being learned.

We have implemented the learning safety net with a combined pre- and post-learning strategy that places strict bounds on what the agent can learn. The two strategies support a variety of behavioral constraints to be specified naturally and computationally efficiently. When the system designer or individual user specifies a constraint through the pre-learning strategy, the resulting MDP explicitly disallows actions from states in which the constraint dictates those actions should never be taken. Thus, no matter what transition probabilities the agent learns, it cannot select those actions. For example, in E-Elves, a pre-learning constraint ensures that the auction MDP does not assign someone to a role when no bids have been received, regardless of the remaining time, by specifically excluding the "assign" action from those states in which no bids have been received.

The post-learning strategy ensures that the MDP has desirable properties after the learning algorithm runs. Rather than exhaustively check every state of every policy generated by the possible MDPs (corresponding to different types of meetings, roles, etc.), our learning algorithm uses heuristics to isolate a small number of states where the (non)existence of the property should be most clearly seen. The algorithm can check these states to determine whether a property holds for a given MDP. For instance, the delay MDP should not learn that all actions are too costly and thus generate a policy of complete inaction. In other words, there should be at least some states in which the agent will take some action other than "wait" or "ask". The heuristic *the user is not at the meeting room 15 minutes after the meeting started* isolates a small number of states where we can expect the delay MDP policy to specify some action (usually delaying the meeting). If we see such an action in the policy, we can immediately stop our search because we know the desired property does indeed hold for the learned MDP. If the post-learning check finds that a required property does not hold, the learning algorithm resets the model parameters back toward the pre-learning values until it finds an acceptable set of values (i.e., ones that have the required properties). This heuristic-based approach is our first step toward tackling the open research issue of providing post-learning guarantees within E-Elves.

5. EVALUATION

We have used the E-Elves system within our research group at USC/ISI for the last six months, with continuous operation 24 hours a day, 7 days a week, since June 1, 2000 (occasionally interrupted for bug fixes and enhancements). There are nine agent proxies (belonging to nine users *dinglei*, *ito*, *jungh*, *kulkarni*, *modi*, *pynadath*, *scerri*, *nair*, and *tambe*), one agent proxy for a project assistant, one capability matcher (with proxy), and an interest matcher (with proxy). Section 2 discusses the key functionality provided by this system. Figure 4 plots the number of daily messages exchanged by the proxies over three months (6/1/2000-8/31/2000). The size of the daily counts reflects the large amount of coordination necessary to manage various activities, while the high variability illustrates the dynamic nature of the domain.

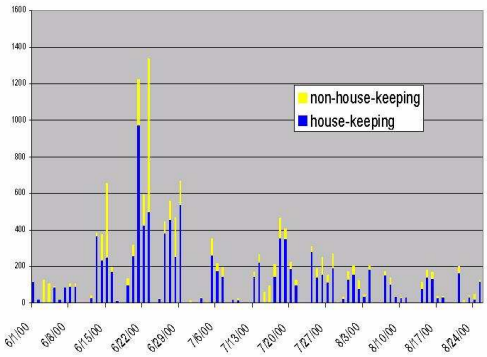
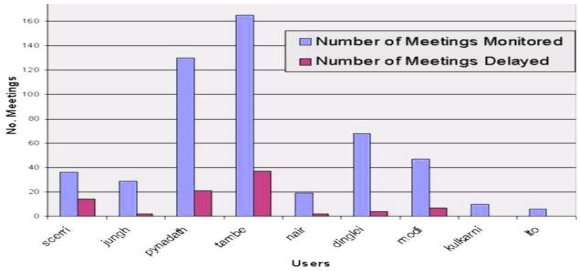
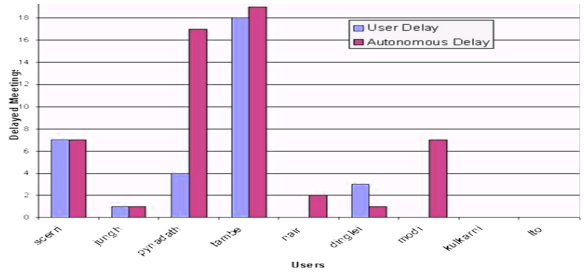


Figure 4: Number of daily coordination messages exchanged by proxies over three-month period.

The general effectiveness of E-Elves is shown by several observations. Since the E-Elves deployment, the group members have exchanged very few email messages to announce meeting delays. Instead, Fridays autonomously inform users of delays, thus reducing the overhead of waiting for delayed members. Second, the overhead of sending emails to recruit and announce a presenter for research meetings is now assumed by agent run auctions. Third, a web page, where Friday agents post their user's location, is commonly used to avoid the overhead of trying to track users down



(a) Monitored vs. delayed meetings per user



(b) Meetings delayed autonomously (darker bar) vs. by hand.

Figure 3: Results of delay MDP's decision-making.

manually. Fourth, mobile devices keep us informed remotely of changes in our schedules, while also enabling us to remotely delay meetings, volunteer for presentations, order meals, etc. We have begun relying on Friday so heavily to order lunch that one local "Subway" restaurant owner even suggested marketing to agents: "*. . . more and more computers are getting to order food . . . so we might have to think about marketing to them!!*".

We now present details underpinning the general observations. Figure 3a illustrates the number of meetings monitored for each user. Over the course of three months (June 1 to August 31) over 400 meetings where monitored. Some users had less than 20 meetings, while others had over 150. Most users had about 20% of their meetings delayed. Figure 3b shows that usually 50% or more of delayed meetings were autonomously delayed. In particular, in this graph, repeated delays of a single meeting are counted only once, and yet, the graphs show that the agents are acting autonomously in a large number of instances. Equally importantly, humans are also often intervening, indicating the critical importance of AA in Friday agents.

Over the course of the past three months, our research group presentations were decided using auctions. Table 1 shows a summary of the auction results. Column 1 shows the dates of the research presentations. While the auctions are held weekly, several weekly meetings over this summer were cancelled due to conference travel and vacations. Column 2 shows the total number of bids received before a decision. A key feature is that auction decisions were made without all 9 users entering bids; in fact, in one case, only 4 bids were received. Column 3 shows the winning bid. A winner typically bid $< 1, 1 >$, i.e., indicating that the user it represents is both capable and willing to do the presentation — a high quality bid. When there was only one such bid, the MDP could confidently choose the winner, otherwise it would wait for user input. Interestingly, the winner on July 27 made a bid of $< 0, 1 >$, i.e., not capable but willing. The team was able to settle on a winner despite the bid not being the highest possible, illustrating its flexibility. Finally, columns 4 and 5 shows the auction outcome. An 'H' in column 5 indicates the auction was decided by a human, an 'A' indicates it was decided autonomously. In four of the six times, winner was automatically selected. The two manual assignments were due to exceptional circumstances in the group (e.g., a first-time visitor).

We performed a number of experiments to verify certain desirable properties of the MDPs. As expected, as the *cost* of asking increased, the number of states in which the agent would relinquish autonomy decreased (Figure 5a). Further, as the *likelihood* of the user replying to an agent increased, so did the number of states where the agent would ask (Figure 5b). The experiments showed that the agent was able to trade off intelligently between the wasted time if the user didn't respond, the value of the information the user could provide, the cost of asking the user, and the likelihood of the user replying.

Date	No. of bids	Best bid	Winner	Method
Jul 6	7	1,1	Scerri	H
Jul 20	9	1,1	Scerri	A
Jul 27	7	0,1	Kulkarni	A
Aug 3	8	1,1	Nair	A
Aug 31	4	1,1	Tambe	A
Sept 19	6	-,-	Visitor	H

Table 1: Results for auctioning research presentation slot.

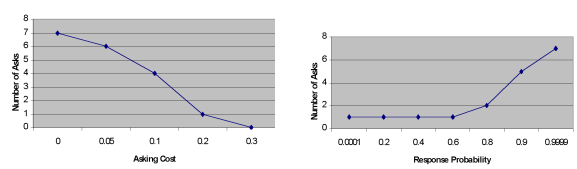


Figure 5: Number of ask actions in policy vs. (a) the cost of asking and (b) the probability of response.

Most importantly, over the entire span of the E-Elves' operation, the agents have *never* repeated any of the catastrophic mistakes that Section 3 enumerated in its discussion of our preliminary decision-tree implementation. For instance, the agents do not commit error 4 from Section 3 because of the domain knowledge encoded in the bid-for-role MDP that specifies a very high cost for erroneously volunteering the user for a presentation. Thus, the generated policy never autonomously volunteers the user. Likewise, the agents never committed errors 1 or 2. In the delay MDP, the lookahead inherent in the policy generation allowed the agents to identify the future rewards possible through "delay" (even though some delays had a higher direct cost than that of "cancel"). The policy described in Section 4.1 illustrates how the agents would first ask the user and then try delaying the meeting, before taking any final cancellation actions. The MDP's lookahead capability also prevents the agents from committing error 3, since they can see that making one large delay is preferable, in the long run, to potentially executing several small delays. Although the current agents do occasionally make mistakes, these errors are typically on the order of asking the user for input a few minutes earlier than may be necessary, etc. Thus, the agents' decisions have been reasonable, though not always optimal. Unfortunately, the inherent subjectivity in user feedback makes a determination of optimality difficult.

6. RELATED WORK

Most of the work done on AA has focused on the interactions

between an individual agent and an individual human, while our research has focused on AA in the team context. For example, we mentioned CAP [9] in Section 3, illustrating that its techniques may face significant difficulties in a team context.

Mixed-initiative systems share responsibility between a human and agent. For example, in TRAINS-95 [3], an agent and a user share a fairly complex planning task, gaining leverage through the abilities of the respective parties. The user addresses aspects involving hard-to-define objective functions, while the agents are assigned problems involving repetitive consideration of detail. In such systems, there is generally only a single user, who is always available, and only a single agent; hence, the AA team decision and AA coordination challenges are avoided.

An aspect of Friday's autonomy reasoning is an assessment of the costs and benefits to the individual user in any transfer-of-control decision. The PRIORITIES system uses decision theory to reason about the costs, benefits and uncertainty associated with alerting a user to the arrival of new email [7]. One of the focuses of the PRIORITIES research is to use Bayesian networks to build a probabilistic model of the user's intentions, so as to better assess the costs involved in interrupting them. PRIORITIES does not, however, consider the team level issues that are considered in E-Elves. In addition, PRIORITIES uses hand-specified costs and values in computing the value of human input, while our approach automatically generates these costs and values from its more abstract MDP planning model.

Much AA research has been motivated by the requirements of NASA space missions [2]. One thread of that research developed an interface layer to the 3T architecture to allow users to take control of an agent at whatever level of control is most appropriate for the given situation. The interface layer approach is fundamentally different from the E-Elves approach as the agent does not explicitly reason about reducing its autonomy.

7. CONCLUSIONS

Gaining a fundamental understanding of AA is critical if we are to deploy multi-agent systems in support of critical human activities in real-world settings like intelligent homes, organizational coordination, electronic commerce, and long-term space missions. Indeed, living and working with the E-Elves has convinced us that AA is a critical part of any human collaboration software. Our initial experience with using a C4.5-based approach to E-Elves provided a negative result that constitutes a useful contribution to AA research. Because of this negative result, we realized that agent teamwork and coordination in such real-world, multi-agent environments as E-Elves introduce novel challenges in AA that previous work has not addressed. In this paper, we have focused on three key challenges: (i) the *AA coordination challenge* requires an agent to avoid miscoordination with teammates, while simultaneously ensuring effective team action; (ii) the *AA team decision challenge* focuses on ensuring effective decisions at the multiple levels of decision-making in teams; and (iii) the *AA safe learning challenge* arises because temporary learning aberrations can have disastrous effects on system users. We discussed several key ideas to address these challenges. In particular, for resolving the *AA coordination challenge*, agents explicitly reason about the costs of team miscoordination, they flexibly transfer autonomy rather than rigidly committing to initial decisions, and they may change the coordination rather than taking risky actions in uncertain states. For addressing the *AA team decision challenge*, AA is introduced at multiple levels of team decision-making hierarchy. For addressing the *safe learning challenge*, we built a considerable amount of domain knowledge into the agents as well as provided boundaries on what the agent could learn at runtime. We have implemented our ideas in the E-Elves system using MDPs, and our AA implementation nows plays a central role in the successful 24/7 deployment of E-Elves in our group. Its success in the diverse tasks of that domain demonstrates the promise that our framework holds for the wide range of multi-agent domains for which AA is critical.

Acknowledgements

This research was supported by DARPA award no. F30602-98-2-0108. The effort is being managed by Air Force Research Labs/Rome site. We thank our colleagues, especially, Craig Knoblock, Yolanda Gil, Hans Chalupsky and Tom Russ for collaborating on the Electric Elves project. Paul Scerri was supported by The Network for Real-Time Education and Research in Sweden (ARTES). Project no. 0055-22, Principal investigator Nancy Reed, CENIIT grant #99.7 and NUTEK grant #1K1P-99-06166.

8. REFERENCES

[1] J. Collins, C. Bilot, M. Gini, and B. Mobasher. Mixed-initiative decision-support in agent-based automated contracting. In *Proc. of Agents'2000*, 2000.

[2] G. A. Dorais, R. P. Bonasso, D. Kortenkamp, B. Pell, and D. Schreckenghost. Adjustable autonomy for human-centered autonomous systems on mars. In *Proc. of the First Int. Conf. of the Mars Society*, 1998.

[3] G. Ferguson, J. Allen, and B. Miller. TRAINS-95 : towards a mixed initiative planning assistant. In *Proc. of the Third Conference on Artificial Intelligence Planning Systems*, pages 70–77, May 1996.

[4] Call for Papers. AAAI spring symposium on adjustable autonomy. www.aaai.org, 1999.

[5] J. P. Gunderson and W. N. Martin. Effects of uncertainty on variable autonomy in maintenence robots. In *Proc. of Agents'99, Workshop on Autonomy Control Software*, 1999.

[6] T. Hartrum and S. Deloach. Design issues for mixed-initiative agent systems. In *Proc. of the AAAI workshop on mixed-initiative intelligence*, 1999.

[7] E. Horvitz, A. Jacobs, and D. Hovel. Attention-sensitive alerting. In *Proc. of UAI'99*, 1999.

[8] V. Lesser, M. Atighetchi, B. Benyo, et al. A multi-agent system for intelligent environment control. In *Proc. of Agents'99*, 1999.

[9] T. Mitchell, R. Caruana, D. Freitag, J. McDermott, and D. Zabowski. Experience with a learning personal assistant. *Communications of the ACM*, 37(7), 1994.

[10] M. L. Puterman. *Markov Decision Processes*. John Wiley & Sons, 1994.

[11] D. V. Pynadath, M. Tambe, H. Chalupsky, Y. Arens, et al. Electric elves: Immersing an agent organization in a human organization. In *Proc. of the AAAI Fall Symposium on Socially Intelligent Agents*, 2000.

[12] S. Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice-Hall, Inc., 1995.

[13] J. R. Quinlan. *C4.5: Programs for machine learning*. Morgan Kaufmann, San Mateo, CA, 1993.

[14] M. Tambe. Towards flexible teamwork. *Journal of Artificial Intelligence Research*, 7:83–124, 1997.

[15] M. Tambe, D. V. Pynadath, N. Chauvat, A. Das, and G. A. Kaminka. Adaptive agent integration architectures for heterogeneous team members. In *Proc. of the ICMAS'2000*, pages 301–308, 2000.