From *Proceedings of the Sixteenth International Conference on Machine Learning*. Bled, Solvenia: Morgan Kaufmann.

Learning User Evaluation Functions for Adaptive Scheduling Assistance

Melinda T. Gervasio and **Wayne Iba** and **Pat Langley**
Institute for the Study of Learning and Expertise
2164 Staunton Court, Palo Alto, California 94306
{gervasio,iba,langley}@isle.org

Abstract

In this paper, we describe INCA, an adaptive, advisable assistant for crisis response. The system lets users guide the search toward particular schedules by giving high-level, operational advice about the solutions desired. Because traces of user interactions provide information regarding the user's preferences among schedules, INCA can draw on machine learning techniques to construct user models that reflect these preferences. We characterize the modeling task as that of learning a weight vector for a linear evaluation function that will lead to the same pairwise preferences between schedules as the user. INCA adapts to individual users by adjusting the weights on its evaluation function using a perceptron-type learning algorithm. To evaluate the system's adaptive capabilities, we designed an experiment involving four types of synthetic users that differed in their evaluation functions and in the level of advice they provide. We present experimental results showing that INCA achieves better performance with more specific advice and with learning, even on users with evaluation functions that have nonlinearities or unobservable factors.

1 INTRODUCTION

In recent years, there has been a growing interest in personalized software as the need has increased for systems that can tailor their behavior to different individuals. For example, the World Wide Web now has electronic commerce sites that suggest new purchases based on a user's past buying behavior, portals that choose banner ads depending on the links a user follows, and travel planners that suggest default choices based on user-specified preferences. Many of these systems use simple heuristics or require a laborious initial phase of user profile construction to achieve personalization. In contrast, an *adaptive user interface* (Langley, 1999) uses traces of its interactions with users to automatically build a model that influences the system's behavior to better suit the user's preferences. In this paper, we present one such system, INCA, an adaptive scheduling assistant that learns user evaluation functions over schedules by extracting preference information from the schedules a user chooses during interaction with the system.

INCA is a mixed-initiative system designed to aid human users in crisis response. The primary focus of our work thus far has been *urgency*, a defining theme of crisis. We have addressed this issue through the use of case-based reasoning techniques to provide initial candidate solutions (Iba et al., 1998), as well as through the application of machine learning to better predict a user's schedule modifications (Gervasio et al., 1998), both of which lead to more efficient construction of high-quality solutions. In this paper, we extend our work on adaptive assistance within a new framework that allows advisable interaction with INCA. By letting users interact with the system at higher levels of abstraction, this new framework lets them more effectively guide the development of a crisis response.

Users provide guidance in the form of *advice* that directs INCA in its search through a highly underconstrained solution space. The advice users give and the choices they make while interacting with the system implicitly define an evaluation function over solutions. For example, if a user directs INCA to modify a solution to improve its value along some dimension and then accepts the modified solution, then the user is indicating a preference for the new solution over the old one. This presents an opportunity to adapt to individual user preferences by examining interaction traces and inducing user models that approximate these preferences. In this paper, we describe an approach that learns evaluation functions over schedules by applying a perceptron-type learning algorithm to pairwise preferences that are drawn from such interaction traces.

We begin by providing an overview of Inca's interactive response process in our application domain of hazardous material incidents. We then discuss the motivations for higher-level interaction and present our new framework for advisable scheduling assistance. In the following section, we lay the groundwork for our learning experiments by presenting Inca's basic learning algorithm and discussing the representation of the examples and the schedule evaluation function. After this, we present our design for an experiment intended to test our hypotheses on the effects of adaptation, given different types of user evaluation functions and different levels of advice. The results, which we discuss in the succeeding section, support our hypotheses that both learning and more specific advice would improve performance. We conclude the paper by discussing related research and summarizing our plans for future work on adaptive assistance.

2 HAZMAT INCIDENT RESPONSE WITH INCA

Inca is a *case-based*, *interactive*, and *adaptive* computational assistant for crisis response. It uses a case library of previous solutions to seed the problem-solving process with an initial candidate plan and schedule, which lets users develop high-quality responses more quickly (Iba et al., 1998). As a mixed-initiative system, Inca benefits from the invaluable input of human users while letting them retain ultimate control over the decision-making process. And by adapting to different individuals, the system can increase the efficiency of the response process by tailoring its assistant behavior to user preferences (Gervasio et al., 1998). In this paper, we focus primarily on the adaptive nature of Inca and secondarily on the new kinds of interaction introduced by the advisable framework.

We have applied Inca to HazMat, a synthetic domain for hazardous materials incidents that we developed using the 1996 North American Emergency Response Guidebook (Transport Canada et al., 1996) and manuals of standard operating procedures for several fire departments. Figure 1 depicts the interactive crisis response process with Inca. Problem solving is triggered by an incident alarm, where an incident consists of a spill and possibly a fire involving some material with hazardous properties such as toxicity, corrosiveness, and flammability. Incidents vary in the type and amount of material involved, the location of the incident, and the characteristics of the spill and any fire. The alarm drives Inca to retrieve a solution to a similar incident from its case library, which

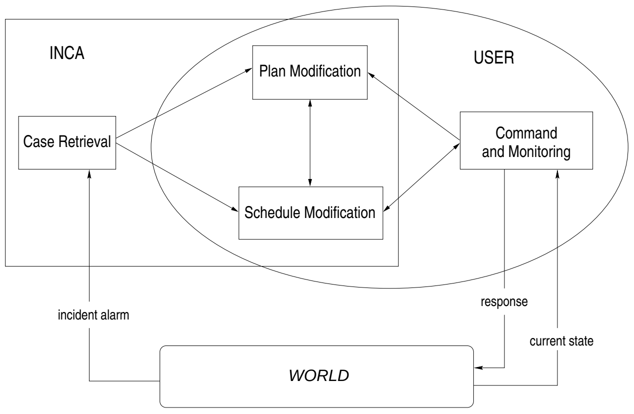

Figure 1: Interactive Crisis Response with Inca

it uses to seed the interactive planning and scheduling phases of crisis response. The primary purpose of planning is to choose a set of actions or jobs, *any* subset of which may be allocated resources during the scheduling phase. Different actions are appropriate for different incidents, with each action addressing particular aspects of a HazMat problem and requiring some minimum set of resources. During scheduling, decisions are made regarding the jobs to schedule and, for each job, the number of resources to allocate, the particular resources to use, a duration, and start time.

Scheduling in Inca is complicated by the fact that actions may be allocated a variable number of resources and assigned arbitrary durations. Allocating more resources to an action will result in greater effects, but at the cost of greater resource consumption that will constrain the other actions and their impact on the situation. For example, assigning all available personnel to combat the fire will extinguish the fire more quickly but will delay stopping the spill, which will result in the release of more hazardous material into the environment. Similarly, scheduling more actions that address a particular aspect of the incident will minimize the negative consequences along that dimension, but possibly at the cost of greater negative consequences along another one. The resulting underconstrained scheduling problem makes automation particularly difficult, calling for an interactive system such as Inca and presenting the opportunity for learning user models that support more productive interaction.

3 ADVISABLE INTERACTION

In previous versions of Inca, a user could only change the retrieved solution at the level of schedule modification operations: adding an action to the schedule, deleting a scheduled action, changing the duration of an action, shifting the start time of an action, or switching an action to a different resource. This type of interaction lets users make the exact changes they desire, but it can also be quite tedious—something of particular concern in time-constrained crisis domains. We decided that more efficient interaction might be achieved if we instead let Inca generate several candidate schedules and have the user choose among them, basically shifting more of the scheduling responsibility to the system. The problem remains, however, that autonomous scheduling is particularly difficult in this underconstrained domain. Our solution was to let the user guide the search for a desirable solution by providing particular types of advice, as dictated and supported by the characteristics of the solution space. This makes Inca an instance of what Myers (1996) calls *advisable planning systems.*

3.1 CHARACTERISTICS OF THE SOLUTION SPACE

A primary characteristic of the solution space for hazardous materials incident response is that it is highly underconstrained. There are many possible solutions for each problem and various ways of evaluating them, with different parameters being more important for particular problems and different users. For example, in a large spill involving a highly toxic chemical in a heavily populated area, the priority in evacuating the inhabitants to safety will probably be on speed rather than cost. In a large fire, where there is a severe shortage of resources, more emphasis may be placed on suppressing the fire near occupied buildings than on stopping the leak.

Although the underconstrained nature of the solution space and the complexity of the objective function make automation difficult, there are also characteristics that suggest a feasible method of navigation. First, we can define a direct relationship between particular parameters of solution quality and the methods for modifying a schedule. Second, we can define a direct relationship between the method for improvement along a particular parameter and the actions themselves. These two properties help shape the kinds of advice Inca can easily operationalize and suggest the framework for our advisable system.

3.2 LEVELS OF ADVICE

The first level of advice involves directing Inca's attention to improve a solution along a particular parameter. We can characterize a solution in terms of its effects on a situation (i.e., its achievement of the problem goals) as well as in terms of its cost (e.g., time taken to reach the goal, number of resources used). In the case of crisis response, as with most real-world problems, we are interested in both—we want to evaluate solutions in terms of their impact on the situation as well as on their cost of execution. However, in our initial implementation, we decided to focus on the former, using our HazMat simulator to project the effects of a solution on the environment.

The simulator tracks the development of the spill, the fire, and the fire and health hazards in terms of the amounts and derivatives of various quantities. By comparing the simulation of a response on the incident with the simulation of an unabated situation, we can come up with a measure of performance improvement along the different parameters. For example, if the amount spilled in the unabated situation was s_u, while the amount spilled with a response was s_a, then the improvement in the amount spilled is $1 - s_u/s_a$.

Thus, an improvement percentage of 1.00 would signify an instantaneous stoppage of the spill, while 0.00 reveals that the response had no effects whatsoever. The simulator keeps track of a total of fifteen parameters, which we can distill into three improvement percentages corresponding to spill improvement, fire improvement, and hazard improvement. At the first level, the user can thus provide Inca with three types of parameter advice: improve spill handling, improve fire handling, or improve hazard handling.

The second level of advice involves indicating the particular modification method to achieve the desired improvement. As stated earlier, there is a direct relationship between the parameter being improved and the schedule modification methods. A choice of improvement parameter constrains the modification methods that may be applied. In our case, the three schedule quality parameters we have defined correspond to the same four schedule modification methods. Specifically, to improve a schedule along any parameter, Inca can either add a new action, assign more resources to a scheduled action, increase the duration of a scheduled action, or start a scheduled action earlier. Applying any of these modifications on a relevant action is guaranteed not to degrade the quality of the schedule with respect to the chosen parameter.

The third level of advice involves specifying the action or job on which to apply the chosen modification method to achieve the desired improvement. An action is relevant to a particular parameter if it affects that parameter. Thus, a given parameter and modification method define a fixed set of actions over which the next decision must be made. For example, if the user directs Inca to improve fire handling by adding an action, then the only relevant choices are those unscheduled actions that affect the fire, such as extinguishment and fire suppression. But if improvement is to be achieved by means of shifting an action earlier, then the only relevant actions are those scheduled actions that do not already start immediately.

The fourth, and final, level of advice involves variable instantiation—that is, specifying particular resources, start times, and durations for a job. Providing advice at this level is equivalent to Inca's original mode of interaction in which users were required to specify the particular schedule modifications they wanted.

3.3 VARIABLE-LEVEL INTERACTION

The basic interaction loop involves the user choosing a schedule from among the current candidates and giving Inca direction on how to improve the solution by providing advice at each level. However, at any level, the user may defer all remaining decisions to Inca by choosing the *ANY* option. With the user's advice, Inca then generates a set of candidates for the next cycle, continuing in this manner until the user is satisfied with one of the current solutions. In keeping with the urgent nature of crisis response, we constrain Inca to a fixed amount of computational resources on each cycle. Thus, more specific advice lets Inca expend more resources in exploring a smaller area of the search space, potentially finding solutions it may not have found with more abstract advice.

In terms of Myers's (1996) three-level taxonomy of advice, parameter specification is a form of *evaluational advice*, which "encompasses constraints on some metric defined for the overall plan." Method specification is a form of *strategic advice*, which "consists of recommendations on how goals and actions are to be accomplished." Finally, action specification and variable instantiation may be seen as forms of *task advice*, which "designates specific goals to be achieved and actions to be performed." As in Myers's work on advisable planning, our objective in incorporating advice taking into Inca was to enable users to more effectively influence the problem-solving process.

4 ADAPTIVE INTERACTION

The traces of user interactions with Inca provide a ready source of training data for adaptation and previous versions of the system used such traces to learn to predict the schedule modification operations a user would choose next (Gervasio et al., 1998). The idea was that, by accurately predicting a user's actions, Inca can facilitate the response development process. We can extend this work within the current framework by using similar inductive learning techniques to predict the particular advice a user will give in the current situation, based on the user's previous behavior.

However, the current advisable interaction framework also provides an opportunity to explore a new learning task. Inca currently employs a weighted linear evaluation function over schedules. If we let c be a particular candidate schedule, and s_c, f_c, and z_c be its spill, fire, and hazard improvement, as defined earlier, then Inca's evaluation function is $I(c) = w_s s_c + w_f f_c + w_z z_c$, where w_p is a weight associated with parameter p. We can thus characterize Inca's task of adapting to a user as that of learning a weight vector $\langle w_s, w_f, w_z \rangle$ that will lead it to prefer (i.e., rank higher) the same schedules as the user. More concretely, if we let the target function (i.e., the user's evaluation function) be U, then we can say that Inca models the user perfectly if $\forall_{c_1,c_2} U(c_1) > U(c_2) \rightarrow I(c_1) > I(c_2)$.

Cast in this manner, Inca's learning task becomes very similar to that of Roger et al.'s Adaptive Route Advisor (1999), which seeks to discover user preferences over alternative driving routes. Drawing from that work, we decided to use a differential perceptron, which learns over sets of examples indicating pairwise preferences—in our case, preferences between schedules. That is, each example is an ordered pair (c_1, c_2), where $U(c_1) > U(c_2)$, and Inca's training task is to find an evaluation function that will minimize the number of examples over which it prefers the wrong candidate.

Without requiring the user to rank every set of candidate schedules, we can still obtain these pairwise preferences from a user's interactions with Inca. In the advisable framework, the user indicates a preference for a particular schedule over all other candidates when he directs the system to improve that schedule. Given that the user has so far had j interactions with Inca in developing a response for a particular problem and now chooses the ith candidate in a set of n candidates, we can extract $n - 1 + j$ pairwise preferences—specifically, $n - 1$ examples capturing the information that the user preferred the ith candidate over the $n - 1$ other candidates in the set, and j examples denoting the user's preference of this ith candidate over all the candidates chosen in the j previous interactions.

Inca uses a batch update scheme, adjusting the perceptron weights once per epoch. Let E be that subset of a set of examples X where I prefers the wrong candidate (i.e., $E = \{(c_1, c_2) \in X | I(c_1) \leq I(c_2)\}$), and let p_c be the value of parameter p for candidate c. Then the differential perceptron update rule will increase the weight associated with p by the amount $\eta_p \sum_{(c_1,c_2) \in E} (p_{c_1} - p_{c_2})$, where η_p is the learning rate associated with p.[1] Intuitively, Inca will move towards the correct preference by increasing the weight for a parameter if that parameter had a higher value for the preferred candidate than for the non-preferred

[1]To speed convergence, we use adaptive learning rates, where each parameter has its own learning rate determined by the current indicated change on its associated weight as well as the direction and magnitude of previous recent changes (Jacobs, 1988).

candidate, and by decreasing it if it had a lower value instead. In standard perceptron training fashion, Inca iterates through its training examples until it makes no more errors or it has gone through more than some maximum number of epochs.

5 ADAPTATION EXPERIMENTS

To evaluate Inca's adaptive capabilities, we developed a variety of synthetic users to test our hypotheses about the effects of evaluation functions and advice on adaptation. While we might expect Inca to adapt to users with evaluation functions similar to its own, we also believe that Inca will be able to tailor its behavior to users with evaluation functions of different forms. Furthermore, we expect that more specific advice during scheduling will, in some sense, provide "better" examples and thus enable Inca to adapt more quickly.

5.1 USER EVALUATION FUNCTION

We designed three types of synthetic users to test our hypothesis on Inca's ability to adapt to different individuals. Type 1 users had evaluation functions of the same form as Inca's weighted linear combination over the spill, fire, and hazard improvement measures. They differed only in the specific weights they attached to each parameter. We created five Type 1 users, each with a randomly generated weight vector. Type 2 users also used a linear evaluation function but this function included three additional features (unknown to Inca) that measured, respectively, the number of scheduled jobs, the number of resources used, and the duration of the schedule. We created three Type 2 users, whose weight distribution varied between the known and unknown features. Specifically, these three users had a weight ratio between known and unknown features of 1:1, 1:3, and 3:1 respectively. Type 3 users used some nonlinear combination of the original three features, and we created three such users with different nonlinear evaluation functions. Specifically, if we let s be the spill improvement, f the fire improvement, and z the hazard improvement, the first Type 3 evaluation function was sfz; the second was $\sqrt[3]{sfz}$; and the third was sf^{1-z}.

5.2 LEVEL OF ADVICE

We created one more user type to evaluate our hypothesis on advice. By default, our synthetic users provide only the highest level advice, namely that of instructing Inca to improve the schedule along *ANY* parameter. User Type 4 was a variant of User Type 1 who differed only in that it gave advice at the parameter level, specifying the parameter along which improvement was desired. Specifically, a Type 4 user requests improvement along that parameter which has the potential to contribute the most to the overall schedule evaluation, given that user's evaluation function. If we let p be the current improvement measure for a particular parameter (spill, fire, hazard), then the amount of potential improvement associated with p is $w_{pu}(1-p)$, where w_{pu} is the user's weight on feature p.

Table 1: Average Number of Interactions, Candidates, and Examples Generated for Different User Types During Problem Solving

User Type	interactions/ problem	candidates/ interaction	examples/ problem set
1	7.58	16.04	757.12
2	5.00	15.77	384.20
3	5.65	18.79	629.67
4	5.86	5.15	219.08
all	6.02	13.94	497.52

5.3 TRAINING REGIMEN

Learning was carried out online using the perceptron training algorithm presented earlier, with random initial weights and the maximum number of epochs set to 100.[2] After each interaction, new examples were extracted from the current set of candidates and the user's choice, and these were were added to the set of all examples. Inca was then retrained on the whole set, starting with the final set of weights from the previous training episode.

To control for the effects of problem sequence on learning and of specific problems on performance, each user was trained five times using five different problem sets, each consisting of a sequence of five randomly generated problems. That is, each user responded to twenty-five different problems broken up into five separate training sequences, with the problems and sequences constant across all users. Also, problem solving was always started with an empty schedule. On average, each response to a problem required 6.02 interactions, with each interaction resulting in an average of 13.94 candidates. Each problem set also generated an average of 497.52 examples over the course of solving all five problems in the set. Table 1 provides a more detailed breakdown by user type. The baselines were determined by running Inca's default evaluation function (which assigns equal weights to the parameters) over the same five problem sets.

[2]The adaptive learning rates η_p were initialized to 0.01 for each parameter p, with $\kappa = 0.1$, $\phi = 0.5$, and $\theta = 0.7$.

Table 2: Predictive Errors for Different User Types, Averaged Over All Users and Problem Sets

User	Baseline		With Learning	
Type	mean	std. err.	mean	std. err.
1	10.59	1.61	4.24	0.40
2	5.49	0.95	6.11	1.08
3	10.48	2.11	6.76	1.20
4	1.12	0.48	0.83	0.43

Table 3: Rank Error for Different User Types, Averaged Over All Users and Problem Sets

User	Baseline		With Learning	
Type	mean	std. err.	mean	std. err.
1	1.93	0.36	0.63	0.08
2	0.82	0.13	0.92	0.15
3	2.34	0.49	1.39	0.26
4	0.07	0.04	0.05	0.03

5.4 PERFORMANCE MEASURES

The objective of adaptation in INCA is to learn to present the user with those candidate schedules that he finds desirable. As discussed earlier, one measure of the quality of a user model is how well its induced preferences over a set of schedules mimic those of the user's. Recall that learning is applied on a set of examples derived from interaction with a user U, with each example being an ordered pair (c_1, c_2), where $U(c_1) > U(c_2)$ and INCA is in error if $I(c_1) \leq I(c_2)$. The *predictive error* of a hypothesis I on a set of examples X, where it makes errors on the subset $E \subseteq X$, is thus defined simply as $|E|/|X|$.

We can measure the effects of learning in an online fashion by testing the current user model after every interaction on the set of new examples generated from that interaction, *prior* to retraining. This achieves the necessary separation between training and testing, since INCA is always evaluated on examples it did not see during learning. If we let e_j be the number of errors INCA makes on the x_j *new* examples generated from interaction j, then we can calculate INCA's predictive error over a problem set involving n interactions as $(\sum_{j=1}^{n} e_j)/(\sum_{j=1}^{n} x_j)$. We can similarly establish the baseline predictive error by summing the errors INCA makes using its default evaluation function and dividing by the total number of examples, the only difference being that, since no training is involved, testing may be done after all the examples are generated.

A second measure of model goodness is based on the rank r of the candidate a user chooses. INCA presents the set of candidates in ranked order, depending on their value according to its current evaluation function. Thus, if INCA had a perfect user model, the user would always choose the highest ranked schedule (i.e., $r = 1$) in the set of candidates the system presents. As before, we can measure the effects of learning in an online fashion by looking at the immediate result of an interaction, prior to retraining. More concretely, let r_j be the rank of the candidate the user chooses at interaction j, where the new candidates were ordered according to the current model. Then INCA's *rank error* over a set of interactions J is defined as $(\sum_{j \in J}(r_j - 1))/|J|$, with perfect models having a rank error of zero. We can again establish the baseline in a similar fashion by using INCA's default evaluation function and summing the relative ranks of the chosen candidates, and then dividing by the total number of interactions.[3]

In summary, our two independent variables were user type (evaluation function) and level of advice, and our two dependent variables for measuring performance were predictive error on pairwise preferences and rank error on the user's chosen candidates. Our two primary hypotheses were that online learning would lead to better performance, and that adaptation would be easier (i.e., performance would be better) on users who provided more specific advice. A secondary hypothesis was that adaptation would be more difficult (i.e., performance would be worse) on users with complex evaluation functions.

6 EXPERIMENTAL RESULTS

The results of our experiment are summarized in Tables 2 and 3, with the error for each user type averaged over all the problem sets and the different users for that type (i.e., twenty-five data points for Types 1 and 4, fifteen for Types 2 and 3). In general, the results support all our hypotheses regarding the benefits of learning and more specific advice. Predictive error (Table 2) was significantly lower with learning for all but User Type 2 (where there was no significant difference), and rank error (Table 3) was also significantly lower except for User Types 2 and 4 (where there were again no significant differences), thus supporting our hypothesis that learning would improve performance

[3]With a variable number of candidates at each interaction, an alternative would be to look at some normalized ranking. However, because of the difficulty of graphically displaying many candidates and the additional cognitive load this presents to human users, we feel that absolute rank is a better measure of performance.

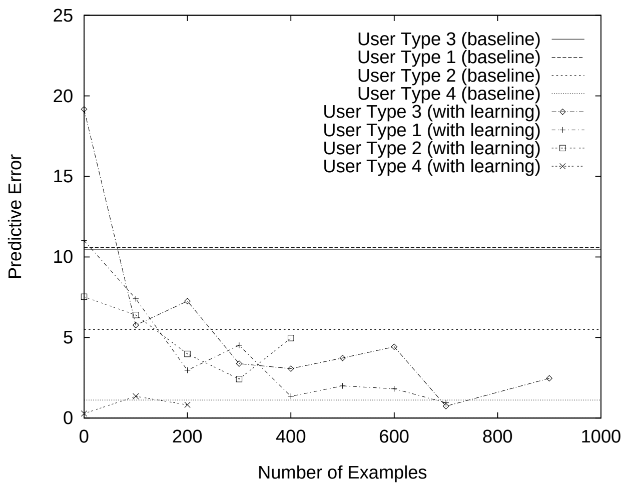

Figure 2: Online Learning Curves Showing Decrease in Predictive Error with Number of Training Examples

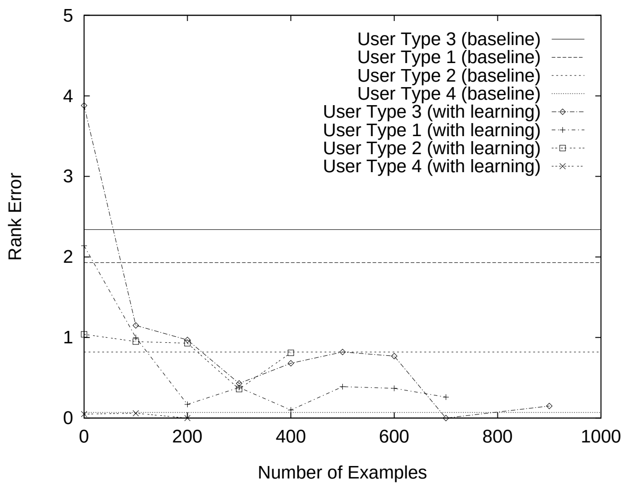

Figure 3: Online Learning Curves Showing Decrease in Rank Error with Number of Training Examples

over the baseline.[4] Errors were also lower when users provided more specific advice (Type 4 vs. Type 1). However, the Type 4 baselines were also much lower, so all we can conclude is that more specific advice simplifies the learning problem and results in better overall performance. The results also support our secondary hypothesis that adaptation will be more difficult on users with more complex evaluation functions (Types 2 and 3).

The results on users with different evaluation functions (Types 2 and 3) provide some insight into the difficulties that can arise when the assumptions of the perceptron algorithm are not met. In particular, target functions with additional features unknown to INCA (Type 2) were more difficult to learn than nonlinear functions using only known features (Type 3). A deeper investigation into the results with Type 2 users reveals that performance with learning was worst with the second user, who gave 75% weight to the unknown features. Performance was best with the third, who gave only 25% weight to these features. This satisfies the intuition that INCA will adapt better to users who do not place too much weight on features unknown to the system when evaluating schedules.

Recall that the results reported in Tables 2 and 3 are averaged over the course of online learning. We expect that, while both types of error may be high early on, they will gradually decrease as more examples are gathered and the system undergoes more training. An alternative, and perhaps more informative, summary of the results are the learning curves shown in Figures 2 and 3, which reveal that, under all conditions, the predictive error and rank error tended to decrease as the number of training examples increased.[5] Consistent with the results presented in Tables 2 and 3, the curves show the least improvement for User Type 2. They also highlight the fact that the baseline errors for User Type 4 were very low to begin with, leaving so little room for improvement that the resulting nonsignificant decrease in rank error was not surprising.

The experimental results generally support our hypothesis that learning will enable INCA to successfully adapt to different users, as shown by a decrease in predictive error of pairwise preferences over all users, and a decrease in the rank error on the user's chosen candidates. In deploying a system such as INCA, we would not want to overwhelm the user with all possible candidates at each interaction, as was done here to avoid confounding factors. Thus, predictive error—which measures the global goodness of a hypothesis—is particularly important, as lower errors along this dimension should translate to greater ability to choose the correct top candidates for presentation to the user. In addition, because schedules are complex entities, we might expect a scheduling assistant to be able to display only one schedule at a time. This makes low rank error particularly important, as it would mean displaying the preferred candidate sooner rather than later, thereby saving valuable time in urgent crisis situations.

[4]All differences presented are significant (paired t-test, $p < 0.05$) unless stated otherwise.

[5]Because different users and problem sets will lead to different numbers of examples, we ended the plot once there were less than 80% of the maximum number of data points available (i.e., we required at least twenty data points over which to average for User Types 1 and 4, and twelve for Types 2 and 3). Using the number of interactions instead of the number of examples leads to learning curves that exhibit similar patterns.

7 RELATED WORK

Much of the earlier work on personalization has been in the realm of computer-aided instruction (e.g., Self, 1974; Clancey, 1979; Sleeman & Smith, 1981). In these systems, detailed student models are built for the purpose of determining level of expertise and selecting lessons to address deficiencies in a student's knowledge while utilizing what he already knows. The system controls the interaction and the goal is to ensure that a student acquires particular skills. In contrast, adaptive computational assistants such as Inca are designed only to aid users during problem solving, the goal being to facilitate the development of a solution while letting users retain control of the process. Inca thus develops user models for the purpose of anticipating a user's choices or identifying a user's preferred solutions. In addition, while the unobtrusive acquisition of user models is highly desirable for computational assistants, tutoring systems are necessarily proactive and can probe specific areas of a student's knowledge to develop better models.

More recently, there has been work on adaptive user interfaces (Langley, 1999)—systems that, like Inca, learn user models for the purpose of modifying their behavior to better reflect user preferences. Most of these systems have focused on the task of recommending items such as news stories (Lang, 1995), music (Shardanand & Maes, 1995), web pages (Pazzani & Billsus, 1997), and movies (Basu et al., 1998) that are consistent with user preferences. Our system differs in that its suggestions are more complex structures (schedules) that entail the generation of candidate solutions aside from rating those solutions. In this sense, Inca is more similar to adaptive interfaces such as Clavier (Hinkle & Toomey, 1994), which suggests configurations for autoclave loading; Hermens & Schlimmer's (1994) form-filling assistant, which suggests default field values for standard forms; and CAP (Dent et al., 1992), a calendar assistant that suggests meeting times and rooms when scheduling appointments. While Clavier uses case-based reasoning techniques to retrieve and adapt a previous solution to solve a new problem, Inca learns an evaluation function over solutions to guide the search for and ranking of candidates for a new problem within a mixed-initiative framework. And while the form-filling assistant and CAP learn separate models for the individual components of a solution, Inca learns a single predictive model over schedules.

As stated earlier, Inca borrows the differential perceptron algorithm used by the Adaptive Route Advisor (Rogers et al., 1999). Aside from the difference in application domain, the two systems vary in their approach towards finding a solution. The Route Advisor always presents an optimal solution according to its current user model, with exploration of the solution space to generate training examples achieved by tweaking the weights (e.g., by placing all the weight on number of turns to generate a route with a minimal number of turns). In contrast, because of the highly underconstrained nature of its solution space, Inca always presents a set of sub-optimal candidates and relies on user advice to guide the search for better solutions, while also extracting training examples based on the candidate a user chooses at each interaction. Inca performs online learning, and there has been a significant body of work on this topic in the computational learning theory community. Much of that work has addressed simplified decision tasks and has focused on proving theoretical bounds on weighted majority algorithms and various extensions (e.g., Littlestone & Warmuth, 1984; Freund & Schapire, 1997), in contrast to our focus on practical performance in complex problem-solving domains. However, we may still be able to gain insight and direction from their results in our continued search for simple, efficient learning algorithms.

Most earlier work on learning for scheduling has focused on improving the performance of autonomous systems by applying learning techniques to generate higher quality solutions (e.g., Eskey & Zweben, 1990; Zhang & Dietterich, 1995; Schneider et al., 1998) or to generate them more efficiently (e.g., Gratch & Chien, 1993). An exception is CABINS (Miyashita & Sycara, 1995), an assistant for job-shop scheduling that acquires optimization preferences by learning the user's preferred repairs on schedules. CABINS is a case-based system that stores repair cases, which it uses to direct an iterative schedule repair procedure in future situations. The user model is thus embodied in these cases, which are gathered during a knowledge acquisition phase wherein the user directs CABINS to try particular repairs, rates the resulting schedule as acceptable or unacceptable, and assigns salience values (i.e., weights) to the case features. In contrast, Inca's user model takes the form of a linear evaluation function over schedules, which the system uses to order the candidate schedules it presents to the user. Inca also extracts its training examples unobtrusively from its interactions with the user during problem solving.

8 FUTURE WORK

The work described in this paper provides some validation of our work on adaptive, interactive systems, but there is much room for further investigation. Inca currently lets the user direct improvement only with respect to the outcome of the solution. We are currently extending Inca's advice-taking capability by incorporating parameters that measure schedule properties, such as the ones employed by User Type 2. This has, in turn, required the identification and inclusion of additional schedule modification methods to achieve improvement along the new dimensions. In providing the user with more options, however, we must be careful not to introduce too many—otherwise, we risk cognitive overload. In addition, increasing the number of parameters complicates INCA's learning task.

We chose to use a simple linear evaluation function and a perceptron-type algorithm because we wanted to achieve fast adaptation. However, as seen in the results with User Type 2, this simplicity may prevent INCA from adapting to user types that are not that unlikely. Using more sophisticated representations and learning methods may solve this problem, but probably at the cost of slower adaptation and infeasible online learning. A possible hybrid approach might use online learning to learn a good approximate user model quickly during problem-solving, and then to use more expressive representations and more powerful learning techniques offline to acquire more accurate models of user preferences.

In this paper, we demonstrated INCA's ability to adapt to a variety of synthetic users but, ultimately, we are interested in its ability to adapt to human users. Thus, we plan to run experiments with actual users to evaluate INCA's adaptation capabilities as well as to better understand the space of real user preferences. An interesting empirical issue is whether linear functions with a small number of features are in fact adequate for useful adaptation, which would obviate the need for more complex representations and learning algorithms. We expect that as we continue to investigate different ways in which computational assistants can adapt to individual users, we will be able to better identify the different kinds of adaptations possible and the approaches that make them feasible.

Another avenue for future work involves taking advantage of user advice to more directly manipulate the weights on the schedule evaluation function. Currently, INCA only trains on the preferences implied by the particular schedule a user chooses from a set of candidates. However, a user's advice to improve the schedule along a particular parameter may also be used to directly increment the weight associated with that feature. We might also let the user indicate a willingness to sacrifice some improvement along a particular parameter, which would lead to decrementing the weight associated with that feature.

INCA is part of an ongoing project to develop adaptive computational assistants for crisis response. We recently replaced the system's automated case retrieval and interactive planning with a new planning module that employs conversational case-based reasoning (Muñoz-Avila et al., in press). In the near future, we also plan to begin development of information-gathering assistants, which would actively seek out information that the user needs to make decisions, and also filter incoming information to present the user with relevant information in a timely fashion. Both revisions open up new opportunities for adaptation, such as tailoring the case retrieval's ranking mechanism to order cases by user preference and learning the types of information a user requests when making particular decisions. Also, although we have studied adaptation within the application domain of hazardous material incidents, we believe that our approach will also prove useful elsewhere. In particular, we expect similar benefits from advisability and adaptation in domains where optimization rather than satisfaction is the key issue, and we are exploring other domains to identify areas that may benefit from adaptation to user preferences.

9 SUMMARY

In this paper, we presented INCA's advisable interaction framework, in which users can give advice at variable levels of specificity to guide the search toward desirable solutions. This new framework provided the opportunity to apply a perceptron-type learning algorithm to adapt the schedule evaluation function to different users. We drew on synthetic users with distinct types of evaluation functions and advice strategies to experimentally investigate INCA's ability to adapt to different individuals. The results support our hypotheses that performance would improve with learning and with more specific advice, as measured by the error in predicting users' pairwise preferences and the rank error on users' chosen candidate solutions. The current interest in personalization, and the trend toward mixed-initiative systems that assist rather than replace human users, present exciting new opportunities for applying machine learning in real-world domains, making it increasingly important to develop and evaluate adaptive, interactive computational assistants such as INCA.

Acknowledgments

This work was supported by the Office of Naval Research under Grant N000014-96-1-1221.

References

Basu, C., Hirsh, H., & Cohen, W. (1998). Recommendation as classification: Using social and content-based information in recommendation. *Proceedings of the Fifteenth National Conference on Artificial Intelligence* (pp. 714–720).

Clancey, W. J. (1979). Dialogue management for rule-based tutorials. *Proceedings of the Sixth International Joint Conference on Artificial Intelligence.*

Dent, L., Boticario, J., McDermott, J, Mitchell, T., & Zaborowski, D. (1992). A personal learning apprentice. *Proceedings of the Tenth National Conference on Artificial Intelligence* (pp. 96–103).

Eskey, M. & Zweben, M. (1990). Learning search control for constraint-based scheduling. *Proceedings of the Eighth National Conference on Artificial Intelligence* (pp. 908–915).

Freund, Y. & Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139.

Gervasio, M. T., Iba, W., & Langley, P. (1998) . Learning to predict user operations for adaptive scheduling. *Proceedings of the Fifteenth National Conference on Artificial Intelligence* (pp. 721–726).

Gratch, J. & Chien, S. (1993). Learning effective control strategies for deep-space network scheduling. *Proceedings of the Tenth International Conference on Machine Learning* (pp. 135–142).

Hermens, L. A. & Schlimmer, J. C. (1994). A machine-learning apprentice for the completion of repetitive forms. *IEEE Expert* 9:28–33.

Hill, W., Stead, L., Rosenstein, M., & Furnas, G. (1995). Recommending and evaluating choices in a virtual community of use. *Proceedings of CHI-95: The ACM SIGCHI Conference on Human Factors in Computing Systems*

Hinkle, D., & Toomey, C. (1994). Clavier: Applying case-based reasoning to composite part fabrication. *Proceedings of the Sixth Conference on Innovative Applications of Artificial Intelligence* (pp. 55–62).

Iba, W., Gervasio, M. T., Langley, P., & Sage, S. (1998). Evaluating computational assistance for crisis response. *Proceedings of the Twentieth Annual Meeting of the Cognitive Science Society.*

Jacobs, R. A. (1988). Increased rates of convergence through learning rate adaptation. *Neural Networks* 1(4):295–308.

Lang, K. (1995). NewsWeeder: Learning to filter netnews. *Proceedings of the Twelfth International Conference on Machine Learning* (pp. 331–330).

Langley, P. (1999). User modeling in adaptive interfaces. *Proceedings of the Seventh International Conference on User Modeling.*

Littlestone, N. & Warmuth, M. K. (1994) The weighted majority algorithm. *Information and Computation* 108:212–261.

Miyashita, K. & Sycara, K. (1995) CABINS: A framework of knowledge acquisition and iterative revision for schedule improvement and reactive repair. *Artificial Intelligence* 76:337–426.

Muñoz-Avila, H., Hendler, J. A., & Aha, D.W. (in press) Conversational case-based planning. *Review of Applied Expert Systems.*

Myers, K. L. (1996) Advisable planning systems. In Tate, A. (Ed.), *Advanced Planning Technology*, (pp. 206–209). Menlo Park: AAAI Press.

Pazzani, M. & Billsus, D. (1997). Learning and revising user profiles: The identification of interesting web sites. *Machine Learning* 27:313–331.

Rogers, S., Fiechter, C.-N., & Langley, P. (1999). An adaptive interactive agent for route advice. *Proceedings of the Third International Conference on Autonomous Agents.*

Schneider, J. G., Boyan, J. A., & Moore, A. W. (1998). Value function based production scheduling. *Proceedings of the Fifteenth International Conference for Machine Learning* (pp. 522–530).

Self, J. A. (1974). Student models in CAI. *International Journal of Man-Machine Studies* 6:261–276.

Shardanand, U. & Maes, P. (1995). Social information filtering: Algorithms for automating "word of mouth". *Proceedings of the CHI-95 Conference.*

Sleeman, D. H. & Smith, M. J. (1981). Modelling student's problem solving. *Artificial Intelligence* 16:171–187.

Transport Canada, the U.S. Department of Transportation, and the Secretariat of Communications and Transportation of Mexico. *1996 North American Emergency Response Guidebook.*

Zhang, W. & Dietterich, T. (1995). A reinforcement learning approach to job-shop scheduling. *Proceedings of the Fourteenth International Conference on Artificial Intelligence.*