Interactive Assessment of User Preference Models: The Automated Travel Assistant

Greg Linden, Steve Hanks, Neal Lesh
Box 352350
Department of Computer Science and Engineering
University of Washington
Seattle, WA 98195-2350
(206) 543-1695 voice, (206) 543-2969 fax
{glinden, hanks,neal}@cs.washington.edu

ABSTRACT

This paper presents the *candidate/critique* model of interactive problem solving, in which an automated problem solver communicates *candidate solutions* to the user and the user *critiques* those solutions. The system starts with minimal information about the user's preferences, and preferences are elicited and inferred incrementally by analyzing the critiques. The system's goal is to present "good" candidates to the user, but to do so it must learn as much as possible about his preferences in order to improve its choice of candidates in subsequent iterations. This system contrasts with traditional decision-analytic and planning frameworks in which a complete model is elicited beforehand or is constructed by a human expert. The paper presents the *Automated Travel Assistant*, an implemented prototype of the model which interactively builds flight itineraries using real-time airline information. The ATA is available on the World Wide Web and has had over 4000 users between May and October 1996.

Keywords: artificial intelligence, user preferences, user modeling, information filtering, information retrieval, utility, dialogue, travel

1. Introduction

Building an accurate user model is essential to decision making and decision-support tasks; a model of the user's preferences is required to make good decisions or to suggest good alternatives. Representations for preference models have been studied extensively in the literature on multi-attribute utility theory (e.g. [Keeney & Raiffa, 1976]), which provides compact representations and elicitation techniques for preference models, but generally assumes that the model is built by a human expert. Problem solvers like AI planning algorithms generally assume that the complete preference model is provided as an input, but this is not a good approach to interactive problem solving in complex domains. Ahead-of-time elicitation demands a tremendous amount of information from the user, most of which will be irrelevant to solving the particular problem at hand. An alternative approach has been to infer a user model automatically over multiple interactions with the user that is used to support decision making and information filtering (e.g. [Thomas & Fischer, 1996], [McCalla et al., 1996], [Mukhopadhyay & Mostafa, 1996]). But, there is also a class of problems for which a user model must be built up quickly and without previous problem-solving episodes, thus requiring the direct participation of the user. Consider, for example, the following interaction between a travel agent and a client:

Client:	"I want to fly from Seattle to Newark next Tuesday afternoon."
Agent:	"I've got a United flight at 3:30pm for $500 and an American flight at 12:30pm for $520."
Client:	"I can't leave before 3:00pm but I do prefer American."
Agent:	"I have another American flight through Denver at 4:00pm for $530."
Client:	"That's pretty expensive. I'd be willing to go on a later flight or another airline if it'd be much cheaper."
Agent:	"The cheapest flight is USAir at 8pm for $490."
Client:	"In that case, the American flight is fine."

Note that as the interaction progresses, the travel agent learns more and more about the client's preferences. The travel agent learns that the client prefers to fly on American Airlines, is somewhat price-sensitive, and has both hard and soft time constraints. Additionally, the client's preferences are rather complex and reflect complicated tradeoffs between cost of the flight, airline, and departure time. Our system aspires to this sort of interaction, where the system provides information about available options and the user provides information about the quality of those options. The system's user model — and thus the quality of the proposed options — improves over time, ultimately resulting in an option that is acceptable to the user. We consider a specific class of these models called *candidate/critique*, in which communication from the system is in the form of *candidate solutions* to the problem, and communication from the user is in the form of *critiques* of those solutions.

The main goal of the system is to present the user with an "acceptable" solution, but to do so, the *candidate critique agent* (CCA) must balance several competing needs. First, the CCA must attempt to display the optimal solution available in the dataset by suggesting optimal and near-optimal solutions based on its current model of the user's preferences. Second, the CCA must try to elicit and refine the user model. This may involve displaying "bad" candidates; the critique of a bad candidate can indicate which attributes are the most important. Third, the CCA must also describe the range of available solutions in the dataset to the user. Note, in the above interaction, that the client does not accept the optimal solution as soon as it is presented but only after he is convinced that it is the optimal solution by being told that the cheapest possible flight is $490.

In this paper, we present a general framework for building candidate/critique agents and we describe the implemented Automated Travel Assistant (ATA) system[1], a CCA for assisting users with planning airline travel that provides real, current information from the Internet Travel Network world wide web service. In a typical interaction with ATA, the user initially provides some preferences over itineraries — perhaps only the departure and destination cities and the approximate dates of travel — and the system provides several

---

[1] Available at http://www.cs.washington.edu/homes/glinden/TravelSoftBot/ATA.html

itineraries that satisfy those preferences. After examining the itineraries offered by the system, the user notes favorable or unfavorable characteristics. The system responds to the user's actions, offering new flight information, and the interaction continues until the user finds a satisfactory itinerary.

Many tasks besides making travel plans fit the candidate/critique model, such as assisting people find information on the Web, selecting merchandise, or graphical layout problems where the person is searching for the layout which best satisfies their preferences.

The contributions of this paper are an exploration of the use of candidate/critique models to elicit and refine user models, the design and implementation of a complete system for performing candidate/critique interactions in a travel domain, and four techniques that are effective in improving these interactions. These techniques are incorporating default preferences into the user model, suggesting trips that lie on the extreme spectrum of available trips (e.g. the cheapest trip), introducing variety into the suggested trips based on a definition of when one trip is *significantly different* than another, and a criterion for determining when one trip is *dominated* by another.

The rest of this paper is organized as follows. In section 2, we give a formal problem specification and discuss the abstract candidate/critique model, then in section 3 we describe the design of our CCA, including the four techniques mentioned above. In section 4, we briefly describe the Automated Travel Assistant system, a prototype CCA, and discuss an extended example of how a typical interaction between a person and our system. In section 5, we discuss related work. In section 6, we discuss future work,. We conclude and summarize in section 7.

2. Problem Specification

In this section, we first discuss the candidate/critique model in general, define key terms and present simplifying assumptions, describe our user model, and specifying the input/output of the CCA agent.

2.1 General Architecture

We ultimately aspire to produce automated decision support systems that produce dialogues like the hypothetical interaction between travel agent and user presented in [Section 1]. In that case, a free-form natural-language dialogue allows solution information to be communicated concisely from the system to the user and allows arbitrary information about the user's preferences to be communicated from user to system.

Although we do not attempt to implement a natural language interface, we would still like to capture the essence of this problem-solving process. In these problems, the system has access to a large dataset and problem-solving methods unavailable to the user. The user has access to preference information not directly available to the system. The basic mode of interaction is iterative and cooperative, where the system and the user both attempt to convey only relevant knowledge. The problem is considered solved when the user is presented with a solution he considers acceptable.

A candidate/critique agent (CCA) implements this style of problem solving but restricts the way in which information is communicated between the agent and the user. The agent presents a short, carefully selected list of candidate solutions to the user. The user responds by either accepting one of these options, or by critiquing one or more of them.

Critiques provide additional information about the user's actual preferences, which in turn lead to new and better candidates.

The format of both candidate solutions and critiques will depend on the details of the particular problem domain. In their most general forms, preferences can amount to an explicit total order over all candidate solutions, and critiques would be a pairwise comparison between two candidates where nothing more could be inferred about the user's preference ordering. The model is intractable and unrealistic in its full generality, however, so we present a common special case below.

We conclude this discussion of the abstract CCA model by posing the question of evaluation: what constitutes a good CCA problem solver? We consider two evaluation criteria. First, the CCA should lead the user to find an acceptable solution quickly. Second, the CCA should lead the user to find a solution that has high quality relative to the user's preferences. This second criterion might be considered controversial because it is only necessary if the user sometimes accepts a low quality solution.

2.2 Terms and Assumptions

In this section, we introduce the particular simplified model of the general CCA architecture. In this model, the domain can be described using attribute/value pairs, preferences can be described using soft constraints over these attributes, and preferences over constraint violations are additive.

Problem domains are commonly defined using a predefined set of attributes A_1, A_2, …, A_n, each of which takes on values from an underlying set $dom(A_i) = \{v_{i,1}, v_{i,2}, \ldots, v_{i,k}\}$. In this case, a *candidate solution* can be described using a tuple of the form $(v_1, v_2, \ldots, v_n)$. For example, in the travel domain, the dataset might describe all currently available flights and each flight might be represented by a set of attributes including the cost of the flight, airline, departure and arrival cities, and time and date of travel.

We describe the user's preferences in terms of soft constraints on the values of attributes. A *constraint* is a function $C_i(v)$: $dom(A_i)$ -> [0,1]. We use the convention that $C_i(v) = 0$ means the constraint is fully satisfied and $C_i(v) = 1$ means the constraint is fully unsatisfied. Values in the open interval represent partial satisfaction of the constraint.

An assumption we make that is not inherent to the candidate/critique model, but does make the problem more tractable, is that the preferences are additive independent [Keeney & Raiffa, 1976], meaning that preferences over the individual attributes do not depend on the level at which the other attributes are achieved. For example, we would assume a person's preferences over price (e.g. the person strongly prefers cheaper flights) do not depend on whether or not the flight is a nonstop or how close to the desired arrival time it lands.[2]

2.3 User Model

The purpose of the user model is to describe a person's preferences over a set of solutions. In the most general form, the preferences can be arbitrary formulas that impose a total order over solutions. Under the assumption of additive independence, we can represent the user's preference over candidate tuples as a weighted sum of constraint functions. The user model consists of a set of constraints and a weighting indicating the importance of each constraint. Formally, the *user model* is a pair $(\{C_1..C_n\}, \{w_1,..,w_n\})$ where C_i is a

---

2 The actual definition of additive independence is slightly stronger, but it does not affect our analysis.

constraint and w_i is the weight, a real number in [0,1], of constraint C_i. A user model provides a partial ordering over all solutions. We will call this the *error* of the candidate solution, which is of the form

$$E((v_1, v_2, \ldots, v_n)) = \sum_{i=1}^{n} C_i(v_i)*w_i$$

Note that assuming additive independence simplifies the model specification, reducing it to *n* soft constraints and *n* weighting coefficients, and simplifies the notion of what a critique is. If the assumption holds, the quality of a candidate solution can be incorrectly computed for only one of two reasons: either the soft constraint for one of the attributes is incorrect, or one of the attributes is weighted improperly. In our implementation in the travel domain, the user is allowed to adjust both of these model parameters directly.

A user model can either completely or partially describe a user's preferences. In our system, the user model initially describes only a few of the users preferences. As weights are adjusted or constraints are added or updated, the user model becomes a more accurate reflection of the user's true preferences.

2.4 Candidate/Critique Interaction

In this section, we describe the interaction between a candidate/critique agent and a user and specify the input/output behavior of the CCA.

On each iteration, the CCA uses the current user model to suggest a set of annotated solutions. In our implementation, a solution is annotated if it has the best value in a particular attribute of all the candidate solutions with respect to the current user model. For example, in the travel domain, the cheapest trip of all trips considered by the system would be labeled as "cheapest". Formally, as shown in [Figure 1], the CCA is a function from a user model and a set of solutions to a small set[3] of suggested solutions. The system calls the CCA with the current user model, and then presents the suggested solutions.

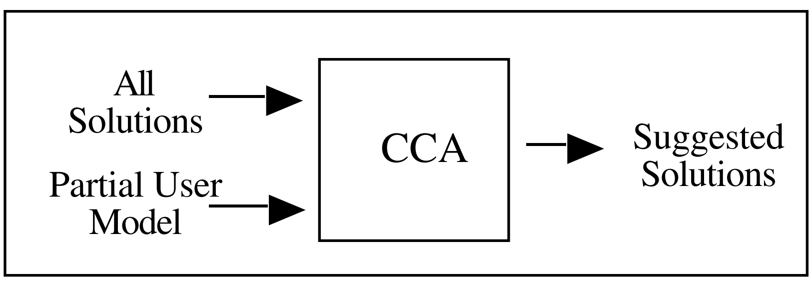

Figure 1: Input/output for the CCA. CCA takes as input all available solutions and a partial user model and generates suggested solutions for the user to evaluate.

After a set of solutions has been suggested the user can either choose one and end the interaction, or add a new constraint, modify an existing constraint, or adjust the weighting of a constraint. This can be accomplished, as in our implementation, though the use of a graphical user interface that allows the user to critique the solutions suggested by the CCA. After the user critiques the suggested candidates, CCA is called with the updated user model, which results in a new set of solutions being suggested.

3 In our implementation, five solutions are suggested in each iteration.

3. CCA Design

In this section we first discuss general design principles of a CCA and then describe four general-purpose techniques for building CCA. In the process of describing these techniques, we show how we instantiated them within the travel domain.

3.1 Design Principles

The overall objective of the system is to help the user find an optimal solution quickly. Hypothetically, presenting the optimal solution for the user immediately would clearly lead to a short, high-quality interaction, but the CCA's user model is only initially a very rough approximation of the user's true preferences and, as such, will not typically generate the optimal solution with respect to the user's true preferences.

In general, there are two ways to generate a short, high-quality interaction: Find a solution that satisfies the user during this iteration of the interaction and present information to the user that increases the likelihood of generating a satisfactory solution in a future iteration. The former can be implemented by optimizing over preferences and requires an accurate user model. The latter involves providing information that allows the user to evaluate the solutions, understand what types of solutions are available, and express additional preferences, refining the user model.

Our technique combines both of these approaches. The system has three goals in terms of presenting solutions to the user:

- Suggest solutions that are optimal or near-optimal with respect to the user preferences
- Inform the user of the full range of available solutions
- Present solutions that allow the CCA to learn more about the user's preferences and update the user model

Note that these three goals can often be in conflict. Suggesting optimal and near-optimal solutions can allow the user to end the interaction very quickly if the user model is accurate, but these solutions may not provide information about the range of options or motivate the user to provide additional preference information. Providing information about the range of available options allows the user to determine what better solutions are available and what the tradeoffs are between solutions, but these solutions are often sub-optimal. For example, in the travel domain, presenting the user with a $300 round-trip from Seattle to Chicago is meaningless unless the user has some information about the range of prices among all flights. Eliciting additional preference information may require presenting "highly critiqueable" solutions that are not optimal. In the next section, we discuss the problem of selecting solutions that satisfy these three goals.

3.2 Algorithm

In this section, we describe the CCA algorithm which uses the partial user model to suggest possible solutions to the user. We describe the CCA by first presenting a simple algorithm and then presenting a series of four improvements to this algorithm.

A straightforward CCA would simply rank all possible solutions according the stated preferences and display the top choices, selecting arbitrarily if the preferences did not provide a total order. For example, in the travel domain, if the user indicated that he wanted to a one-way flight from Seattle to Newark on January 2, this CCA would arbitrarily choose a few of the hundreds of available flights between those two cities on that day. If the user then stated a preference for cheaper flights, this CCA would display a few of the cheapest flights.

Will this approach work? As long as the user continues to state and refine constraints that reflect his true preferences, the user model will eventually converge to an accurate model of the user's preferences. Thus, the CCA will eventually suggest the solution that optimally satisfies the users preferences. However, we believe this approach would require the user to go through many unnecessary iterations with the CCA. Furthermore, a user might not state all of his preferences and may not have enough information about available solutions if he is not presented with more varied or extreme information. We have incorporated the following four improvements over the straightforward approach described above.

1. Add default preferences to model:

The CCA adds default preferences to the user's expressed preferences, generating a user model that is likely to reflect more accurately the user's true preferences. In our implementation in the travel domain, the default preferences are currently that the user is moderately price sensitive, prefers fewer stops to more stops, and prefers to fly on as few different airlines as possible. For other attributes, if the user has not specified a preference over that attribute, he is assumed to be indifferent. Adding default preferences saves the user work by allowing him to provide fewer preferences initially. Assuming the defaults are accurate, the CCA will generate a shorter interaction since the user is not required to explicitly provide all the preferences.

2. Exclude dominated solutions:

The CCA should never suggests a solution that is *dominated* by, or strictly inferior to, another suggested solution. For example, if the user prefers cheaper flights and has expressed that he has no preference over airlines, a \$49 United flight is better than a \$59 Continental flight that leaves at the same time (and is equal in other respects, as well). Even if the Continental flight is the second best trip, there is no need to show it to the user since he should strictly prefer the United flight. Formally, a solution S_2 is dominated by another solution S_1 if

$\forall$ constraints C_i in the user model:

$$C_i(v_{i,1}) \leq C_i(v_{i,2})$$

and for some constraint C_j:

$$C_j(v_{j,1}) < C_j(v_{j,2})$$

where $v_{i,1}$ and $v_{i,2}$ are the values of the ith attribute of S_1 and S_2 respectively.

3. Prefer significantly different solutions:

The CCA will not suggest solutions that are too similar to other suggested solutions. Formally, a solution is significantly different than another solution if

$$\sum_i w_i \left| v_{i,1} - v_{i,2} \right| \geq \delta$$

where $v_{i,1}$ and $v_{i,2}$ are the values of the attributes for the first and second solutions[4] and w_i is a weighting of the difference for the attribute, and δ is the difference threshold.

This criterion biases the selection process in favor of variety. For example, if the system is picking two of a set of three flights, a 8am United flight, a 2pm United flight, and

[4] Non-numeric attributes (e.g. airlines) with different values will be considered maximally different. For example, "Delta" and "United" would be considered maximally different

a 2pm Delta flight (all other attributes being equal), selecting the 8am United and 2pm Delta gives the most variety and the most information to the user. Variance in the set of solutions selected by the CCA allows the user to eliminate entire classes of solutions. For example, showing one trip in the morning and one in the afternoon is likely to elicit a user's preference over time of day, if one exists.

4. Suggest extrema:

Extrema are solutions that optimize one attribute of the solution. For example, with air travel, the cheapest possible trip optimizes the price attribute of the trip. Extrema can elicit more information from the user about the relative weighting of their preferences and provide the user with critical information about how much a potential solution could be improved in terms of a specific attribute, given the available solutions and the current preferences. For example, if the system advises the user that the cheapest possible trip is only $20 cheaper, the user may decide that flying on their preferred airline is worth the slight increase in cost.

In our implementation, we find and present two extrema of interest, the cheapest trip and the best nonstop trip. The cheapest trip is defined as the feasible trip[5] that minimizes the price of the trip. When multiple feasible trips all have the minimal price, the one with minimal error (relative to the current user model) is selected. The best nonstop is defined as the trip of all the feasible trips consisting of all nonstop flights with minimal error.

4. Extended Example

We have implemented a CCA, the Automated Travel Assistant (ATA), in the travel domain. The system is available as a web service[6] and has had over 4000 users between May and October 1996, and has been highly regarded by the major Java applet indexing services[7]. To demonstrate the operation of the system, we provide an example of using the system to find a round-trip between the San Francisco bay area and Philadelphia.

The interaction with the system starts with the user providing a minimal amount of information. In this case, the user states that he wishes to travel between San Jose and Philadelphia, leaving any time on September 25 and returning any time on October 6.

The system converts these preferences into a user model, adding default preferences for unspecified attributes: moderate price sensitivity, preference for fewer stops, and a weak preference for flying on fewer different airlines. These preferences are common to almost all users, though the system can revise them if the user's preferences are atypical.

The system finds flights that satisfy the given preferences, groups the flights into trips, and ranks the trips using the preferences in the user model. Of the top-ranked trips, three significantly different, undominated trips[8] will be displayed along with two extrema, the cheapest trip and best non-stop trip.

ATA displays the following:

---

5 A feasible trip is any trip that at least partially satisfies all the preferences.

6 The system is written as a Java applet and is available at the URL http://www.cs.washington.edu/homes/glinden/TravelSoftBot/ATA.html

7 Rated "Top 1% of Java applets" by the Java Applet Review Service (http://www.jars.com) and "Featured Applet" and "What's Cool" by the Gamelan service (http://www.gamelan.com).

8 *Significantly different* and *dominated* are defined in [Section 3.2].

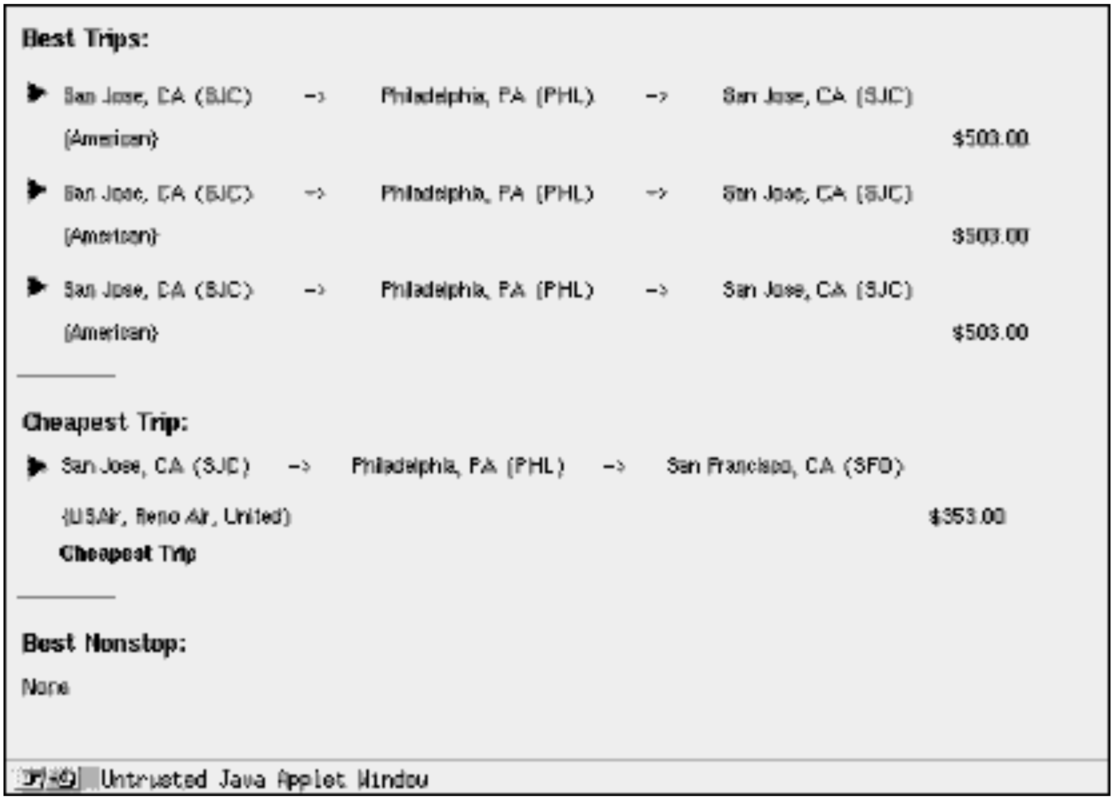

Figure 2: Trips displayed by ATA after the initial query for a round-trip from San Jose to Philadelphia leaving any time Sept. 25 and returning any time Oct. 6. Each trip can be expanded to show information about the flights of the trip and each flight can be expanded to show information about the flight legs.

In this case, no non-stop trips are available between San Jose and Philadelphia. The user is interested in seeing a non-stop trip. He's largely indifferent between flying out of San Francisco and San Jose airports. He modifies his expressed preferences accordingly, setting the value of San Francisco to be slightly lower than San Jose to express a mild preference for San Jose.

Given the additional option of leaving and arriving in San Francisco, the system was able to find non-stop trips. The first trip listed is a USAir nonstop round-trip between San Francisco and Philadelphia.

The user is happy with a non-stop out of San Francisco, but would prefer to fly on United. He adds a preference for United Airlines.

The system considers this new information and offers a few United non-stops. The first trip listed is a United nonstop from San Francisco to Philadelphia round-trip leaving at 11:05am on September 25 and returning at 10:00am on October 6.

The 11:05am flight of the trip leaves too early for the user. The user modifies his departure time for that flight, specifying that he is indifferent between any time 2-5pm, will leave no earlier than about 12pm and no later than about 8pm.

The system finds trips to satisfy these additional constraints, displaying a United non-stop trip with a flight leaving SFO at 2:10pm on September 25 and returning at 10:00am on October 6. Satisfied, the user ends the session.

5. Related Work

Several systems automatically infer a relatively simple user model for information filtering and classification tasks based on observation of the user (e.g. [Burke, Hammond, & Young, 1996] among others), including work using neural networks to learn user models (e.g. [Karunanithi & Alspector, 1996]). Our approach infers a more complex model, but requires direct participation of the user. On the other extreme, decision-support systems that contain hand-coded utility functions can allow more expressive representations than our system, including relaxing the assumption of additive independence, though these systems require extensive effort on the part of a human expert to build the user model.

[Raskutti & Zukerman, 1994] use an approach in the RADAR system where the system requires travelers to disambiguate fully their expressed preferences before querying a fictional database. This approach forces the user to specify all his preferences before he is given any information. In contrast, a candidate/critique session, as implemented in our system, allows the user to retrieve information even if information about user preferences is incomplete.

[Burke, Hammond, & Young, 1996] use an approach to interactions in their FindMe system that is similar in spirit to the ATA system. However, FindMe uses feature vectors to represent data and simple preference representations, severely limiting the expressiveness relative to our work. Due to their limited expressiveness, the domains for which the FindMe system was used have simpler objects and preferences than the travel domain. In addition, the FindMe system does not present extrema or attempt to find significantly different trips.

The TRAINS system, as described in [Allen et. al, 1994] and [Ferguson, Allen, & Miller, 1995], interacts with a user to manage a railway transportation system. As with our ATA, TRAINS was designed to handle a surplus of largely irrelevant information and minimize the amount of information presented to the user. Unlike our work, work on TRAIN does not focus on eliciting a complex model of the user's preferences to guide the search through a large solution space but instead focuses on collaboratively repairing a single solution to the given problem.

In Globe-Trotter [Bose, Biaswas, & Padala, 1989], the travel planner generates an interaction between the system and the user, attempting to emulate the interaction with a real travel agent. Expressed user preferences are progressively refined as the system provides information and the user modifies his profile. Globe-Trotter matches incomplete information about user preferences to a set of predefined cases that fully express the preferences. [Huang & Miles, 1995] use a similar approach. However, this approach "makes (often unwarranted) assumptions based on the user-provided information," as [Cleary & Zeleznikow, 1991] argue, and can direct the user toward inappropriate, rigid, predefined profiles of user preferences. Our approach makes very few assumptions about the user, instead providing the user with information about potential options while seeking more information from the user about his preferences.

Several on-line air travel planning systems currently exist that access real-time airline data, including the Internet Travel Network (http://www.itn.net), Travelocity (http://www.travelocity.com), and PCTravel (http://www.pctravel.com). These systems often provide a large list of undifferentiated information, forcing the user to sort through and find the pieces of relevant information in the sea of irrelevant data. Additionally, it is often difficult and tedious to modify the expressed preferences once flight information is retrieved from the system.

6. Future Work

As noted in [Section 2.2], our implementation relies on the additive independence assumption. Although our experience has been that, in most cases, the assumption is warranted, there are some exceptions. For example, one user had a strong aversion to stopovers only if they were in Chicago. We will work toward extending the specifics of the CCA model to handle preference structures that violate the additive independence assumption at least in limited instances. The challenges here are to recognize an assumption violation during the problem-solving session itself and developing interfaces that support critiquing options given the richer preference models.

We intend to run a series of user tests to analyze the impact of each of the four techniques presented in [Section 3.2] and to show that they generate shorter, higher-quality interactions than simpler approaches. As part of this evaluation, we will compare our implemented ATA system to both the simplified candidate/critique agent discussed in [Section 3.2] and to the existing on-line travel services that do not use an interaction-based approach.

Assumptions made by adding default preferences represent *non-shared beliefs* — information known and used by the system but not communicated to the user. Ideally, the system would share this information with the user. In particular, we intend to extend the system so that it explains why the trips displayed were displayed by providing information about the how the trip was ranked.

We intend to extend the system with a knowledge base that provides advice to the user on his travel itinerary. In our current ATA system, we have assumed that the expressed user preferences are valid. However, there are trips that technically violate at least one of the constraints but may still be of interest to the user. For example, if the user has stated that he wants to fly on November 9 but a significantly cheaper flight leaves on November 8, the system may want to advise the user of the cheaper flight and offer to change the preference to allow flights on November 8. The knowledge base will include simple, well-known rules such as staying over a Saturday night as well as more sophisticated rules that require the system to search for better trips across a broad set of solutions.

7. Conclusion

In decision-making and decision-support tasks, a model of the user's preferences is required to make good decisions or to suggest good alternatives. We explore the use of candidate/critique models to elicit and refine user modes. In the class of interactions considered in this work, the candidate/critique agent (CCA) has complete information over a large dataset but incomplete information about user preferences. The user has complete information about his preferences but incomplete information about the dataset. The CCA attempts to find and present the optimal solution with respect to the unknown user preferences. On each iteration of the interaction, the system presents a small set of solutions to the user and the user expresses additional preference information based on these solutions. The interaction ends when the user accepts one of the suggested solutions.

We present a framework for developing candidate/critique agents (CCA's) and a complete implementation of a CCA in the travel domain. We offer four general techniques for improving an interaction between a CCA and the user: default preferences, presenting extrema, presenting significantly different solutions, and eliminating dominated solutions.

Our implemented CCA, the Automated Travel Assistant, assists the user in finding an optimal or near-optimal trip by presenting the user with carefully selected exemplars that characterize the solution space and allowing the user to express additional or modify existing preferences. The system is available on the World Wide Web and has had over 4000 users between May and October 1996.

References:

Allen-J et al., "The TRAINS Project: A case study in building a conversational planning agent", Journal of Experimental and Theoretical AI, pp. 7-48, 1995.

Bose-P-K & Biswas-G, Rao-Padala-A-M, "Globe-Trotter: An intelligent flight itinerary planner", IEEE Expert vol.4, no.2, pp. 56-64, Summer 1989.

Burke-R, Hammond-K, Young-B, "Knowledge-Based Navigation of Complex Information Spaces", AAAI 1996.

Cleary-D & Zeleznikow-J, "L-CATA: An intelligent logic based expert travel assistant", Eleventh International Conference on Expert Systems and their Applications, pp. 111-22, May 1991.

Ferguson-G, Allen-J, & Miller-B, "TRAINS-95: Toward a mixed-initiative planning assistant", Proceedings of the Third Conference on Artificial Intelligence Planning Systems (AIPS), May 1996.

Huang-Y & Miles-R, "Combining case based and constraint based techniques in travel reservation systems", Proceedings of the 11th Conference on Artificial Intelligence for Applications, pp. 46-54, Feb. 1995.

Karunanithi-N & Alspector-J, "A feature-based user model for movie selection", Proceedings of the Fifth International Conference on User Modeling, pp. 29-34, 1996.

Keeney-R & Raiffa-H, *Decisions with Multiple Objectives: Preferences and Value Tradeoffs*, Wiley, 1976.

McCalla-G, Searwar-F, Thomson-J, Collins-J, Sun-Y, & Zhou-B, "Analogical user modelling: A case study in individualized information filtering", Proceedings of the Fifth International Conference on User Modeling, pp. 13-20, 1996

Mukhopadhyay-S, Mostafa-J, & Palakal-M, "An adaptive multi-level information filtering system", Proceedings of the Fifth International Conference on User Modeling, pp. 21-28, 1996

Raskutti-B & Zukerman-I, "Generating queries during cooperative consultations", Proceedings of the 6th Australian Joint Conference on Artificial Intelligence, pp. 389-94, Nov. 1993.

Raskutti-B & Zukerman-I, "Acquisition of information to determine a user's plan", ECAI 94 pp. 28-32. August 1994.

Thomas-C & Fischer-G, "Using agents to improve the usability and usefulness of the world-wide web", Proceedings of the Fifth International Conference on User Modeling, pp. 5-12, 1996

Wilson-M & Borning-A, "Hierachical constraint logic programming," Journal of Logic Programming, vol. 16, no. 3-4, pp. 277-318, Jul-Aug 1993.