Agile Preference Models based on Soft Constraints

Boi Faltings
Artificial Intelligence Laboratory
Ecole Polytechnique Fédérale de Lausanne (EPFL)
CH-1015 Lausanne, Switzerland
boi.faltings@epfl.ch

Pearl Pu, Jiyong Zhang
Human Computer Interaction Group
Ecole Polytechnique Fédérale de Lausanne (EPFL)
CH-1015 Lausanne, Switzerland
{pearl.pu, jiyong.zhang}@epfl.ch

Abstract

An accurate model of the user's preferences is a crucial element of most decision support systems. It is often assumed that users have a well-defined and stable set of preferences that can be elicited through a set of questions. However, recent research has shown that people very often *construct* their preferences on the fly depending on the available decision options. Thus, their answers to a series of questions before seeing decision options are likely to be inconsistent and often lead to erroneous models. To accurately capture preference expressions as people make them, it is necessary for the preference model to be *agile*: it should allow decision making with an incomplete preference model, and it should let users add, retract or revise individual preferences easily. We show how constraint satisfaction and in particular soft constraints provide the right formalism to do this, and give examples of its implementation in a travel planning tool.

Introduction

Understanding and representing a user's preferences accurately is a crucial element of most decision support systems. We consider in particular decision support systems that help people find the most preferred outcome in a large set of possibilities. A typical example is a configuration system that generates a large number of feasible solutions that must satisfy the requirements. If the set of possibilities is very large, users cannot compare them visually, so an automated filter is required that finds the most preferred ones. Such filters rely crucially on an accurate model of preferences.

We consider that preference are ephemeral and likely to change between different sessions with a decision aid tool. For example, in a travel planning system, preferences for times, airlines, etc. all depend on the particular scenario. Thus, we consider that the preference model needs to be constructed specifically for each scenario at hand. Since users are not willing to spend a lot of time on this process, this can become a significant challenge.

Most existing systems *elicit* preferences through a series of questions whose answers precisely define the user's preferences. For example, a travel planning tool such as Travelocity asks each user several questions about the itinerary and time and airline preferences, and then returns a set of possible choices based on the resulting preference model. Certain e-commerce sites go further and lead the user through a fixed sequence of questions that determine the final choice. Elicitation through questions is the method proposed in classical decision theory (Keeney & Raiffa 1976) and research continues on improving its performance (Boutilier *et al.* 2004). Such elicitation processes implicitly assume that users have a pre-existing and stable set of preferences.

Behavioral decision theory (Payne, Bettman, & Johnson 1993; Carenini & Poole 2002) has studied actual decision makers' behavior. Many years of studies have pointed out the adaptive and constructive nature of human decision making. In particular, a user's preferences are likely to change as a result of the elicitation process itself, so that the answers given to subsequent questions are often inconsistent. Specifically, the following situations can lead a rigid preference elicitation procedure to an incorrect model:

- Users are not aware of all preferences until they see them violated. For example, a user does not think of stating a preference for intermediate airport until a solution includes a change of airplane in a place that he dislikes. This can not be supported by the decision tool that requires preferences to be stated in a predefined order.
- Elicitation questions that do not concern the user's true objective can force him to formulate *means objectives* corresponding to the question. For example, in a travel planning system suppose that the user's objective is to be at his destination at 15:00, but that the tool asks him about the desired departure time. The user might believe that the trip necessarily involves a plane change and take about 5 hours, and thus forms a means objective to depart at 10:00 to answer the question. However, the best option might be a new direct flight that leaves at 12:30 and gets there at 14:30. This solution would not be found using the elicited preference model. This phenomenon has been studied by Keeney(1992) in his work on value-focussed thinking.
- Preferences are often in contradiction and require the user to make tradeoffs. In tradeoffs, users add, remove or change preferences in an arbitrary order. Again, this is not supported by a tool with a rigid preference elicitation procedure.

To support these properties of human decision making, we

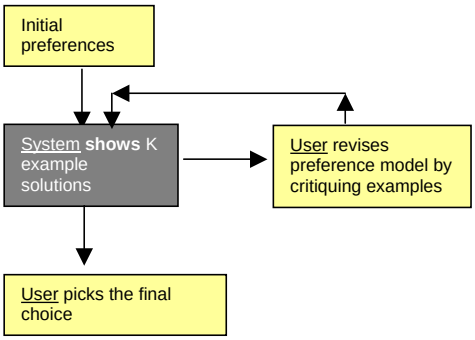

Figure 1: *Example critiquing interaction. The dark box is the computer's action, the other boxes show actions of the user.*

require a preference model to be *agile*: able to support incremental construction and revision of a preference model by the user. More precisely, we define agility as satisfying the following two requirements:

- allow decision making with an incomplete preference model that only captures those preferences that have actually been expressed by the user, and
- allow to add, retract and revise preferences in any order during the decision process.

The need for such preference models has been pointed out earlier, for example by (Ha & Haddawy 1997).

Furthermore, preference construction must be supported by feedback indicating the influence of the current model on the outcomes. A good way to implement such feedback is to structure user interaction as a mixed-initiative systems (MIS). MIS are interactive problem solvers where human and machine intelligence are combined for their respective superiority (Allen *et al.* 1994; Horvitz 1999). MIS are therefore good candidates for such incremental decision systems. A good way to implement a mixed-initiative decision support system is *example critiquing* interaction (see Figure 1). It shows examples of complete solutions and invites users to state their critiques of these examples. This allows users to better understand the impact of their preferences. Example-critiquing as an interface method has been proposed by a variety of researchers, for example (Burke, Hammond, & Young 1997; Linden, Hanks, & Lesh 1997; Shearin & Lieberman 2001).

Modeling Preferences for Example-Critiquing

Modelling the outcome space

In a multi-attribute decision problem, outcomes are modeled using a set of *attributes*. Attributes can be component or feature attributes. A component attribute is a physical aspect of a choice (e.g., the screen size of a PC), while a feature attribute is an abstract concept such as the suitability of a product for outdoor use (good, bad). Formally, an attribute can be modelled by a *variable* that can take values in a fixed *domain*. An outcome can be modelled as a combination of attribute-values. There may be *constraints* that rule out certain combinations of attribute values.

A constraint satisfaction problem (CSP) is characterized by a set of n variables $X_1, ..., X_n$ that can take values in associated discrete domains $D_1, .., D_n$, and a set of m constraints $C_1, .., C_m$, which we assume to be either unary (one variable) or binary (two variables), and which define the tuples that are allowed for the variable or pair of variables. Solving a constraint satisfaction problem means finding one, several or all combinations of complete value assignments such that all hard constraints are satisfied. Thus, the space of possible outcomes in a decision support system can be modelled as the solution space of a constraint satisfaction problem.

Modelling preferences

People make decisions by considering a set of *criteria* that involve these attributes. For example, a criterion could be how tiring a trip is likely to be, or how well its schedule fits the tasks to be accomplished. Each criterion is a function of one or several attributes. Many criteria are simple functions of a single attribute: for example, whether the arrival time is early enough for a 18:00 meeting, or whether the airline fits the user's preference. Others can be more complex: for example, how tiring a trip is depends on the total travel time, the departure and arrival times, the number of stops, etc.

Because preferences naturally arise on criteria, changes in preference models arise as addition, removal or change of importance of threshold for some criteria. In particular, critiques in the example-critiquing model can be formulated in this way.

In classical decision theory, outcomes are ranked according to a utility function formulated on attributes (Keeney & Raiffa 1976). For example, a utility function on 3 attributes a_1, a_2 and a_3 could be specified as:

$$\begin{aligned} U(a_1, a_2, a_3) \quad = \quad & w_1 f_1(a_1) + w_2 f_2(a_2) + w_3 f_3(a_3) + \\ & w_{12} f_{12}(a_1, a_2) + w_{13} f_{13}(a_1, a_3) + \\ & w_{23} f_{23}(a_2, a_3) + w_{123} f_{123}(a_1, a_2, a_3) \end{aligned}$$

The functions $(f_1, f_2, f_3, ...f_{123})$ are called value functions, and each function maps the values of one or a set of attributes to a real number representing their utility. For most decision makers, not all terms need to be present. For example, each attribute might make an independent contribution to the utility of an outcome and in this case only the first three terms would be present. Note that if there are several criteria on the same attributes, they would be combined into a single function. When certain independence assumptions hold, decision theory gives procedures where the weights w and the functions f can be determined based on systematic queries to the decision maker. However, the problem with this framework is that several criteria might be part of the same value function. When criteria change, the elicitation process has to start over again. Thus, it is hard to construct an agile decision support tool based on this mechanism.

Modeling preferences as soft constraints

Besides hard constraints (also known as feasibility constraints) that can never be violated, a CSP may also include soft constraints. These are functions that map any potential value assignment to a variable or combination of variables into a numerical value that indicates the preference that this value or value combination carries. Solving a CSP with soft constraints also involves finding assignments that are optimally preferred with respect to the soft constraints.

Note that soft constraints correspond exactly to criteria that are the basic element of human preference models. Addition and removal of criterion can be easily modelled as adding and removing a corresponding soft constraint. Thus, the soft constraint formalism offers a natural way to implement agile preference models. For example, in a travel planning system, the user might initially start with a single criterion involving arrival time that fixes it to be early enough for the meeting at 18:00. Later on, when he considers arrangements in more detail, he may additionally prefer an arrival that also avoids the rush hour between 16:00 and 19:00. In a preference model consisting of soft constraints, such an additional preference can be integrated by simply adding another soft constraint. By comparison, in a strict utility function, it would require re-evaluating the coefficients of the utility function.

There are various soft constraint formalisms that differ in the way the preference values of individual soft constraints are combined (combination function). For example, in weighted constraint satisfaction(Schiex, Fargier, & Verfaillie 1995), the optimal solution minimizes the weighted sum of preferences. In fuzzy constraint satisfaction(Ruttkay 1994), the optimal solution maximizes the minimum preference value. In our systems, we have almost always used the weighted model as we found it corresponds best to users' intuition about the compensatory nature of tradeoffs. Thus, in the following we assume the weighted constraint satisfaction model.

Soft Constraints for Agile Preference Models

We apply the constraint satisfaction framework to multi-attribute decision problems by formulating each criterion as a separate constraint. Note that using the weighted CSP model, we are able to express the general format of a utility function in classical decision theory, where each of the value functions is a soft constraint.

The constraint programming framework replaces preference *elicitation*, a process of asking specific valuation questions, with preference *construction* where users can manipulate individual criteria as separate constraints. In particular, it satisfies two important goals of agile preference models:

1. it allows users to formulate their criteria precisely without being restricted by the vocabulary of a database schema or elicitation tool,
2. criteria can be added, removed or changed in any order they choose, as the model is not sensitive to an ordering among the preferences.

Furthermore, we will see below that the constraint model supports a variety of tradeoff strategies.

The following example illustrates the difference of a constraint-based preference model with a value function. Suppose that the task is to support a buying decision for a new car. One of the attributes is color and let us assume it has possible values red, green and black. For eliciting a value function, we would ask the user a set of questions (at least 3) that determine the utility values of each of the colors and thus obtain the function f_{color}:

Color	red	green	black
f_{color}	1	-1.0	2.0

In a constraint-based model, a user might first say that his preferred color is red, resulting in a soft constraint such as this:

Color	red	green	black
P_1	1	0	0

Later on, he might discover that black is also available and that actually black is really even better. This would give rise to another soft constraint P_2:

Color	red	green	black
P_2	0	0	2.0

Finally, the system may at some point propose green, which might not be desirable at all and give rise to this preference:

Color	red	green	black
P_3	0	-1.0	0

When ranking the outcomes, the effect of having the value function f_{color} and that of summing the three preferences P_1, P_2 and P_3 is exactly the same. However, the individual preferences could have been stated in any order and at any time in the decision process, rather than having to all be stated in the beginning. Furthermore, they can be retracted or changed individually, resulting in much greater flexibility. For example, if the user finds later in the process that for some reason red is actually not desirable, he can retract P_1 or reduce its weight as a separately identifiable criterion.

There are also two other advantages that turn out to be very important in practice. The first is that options that are not desirable will most likely never come up, and their preference value thus never have to be elicited. This happens frequently because the set of criteria that people are willing to consider in a decision is usually much smaller than the (very large) number theoretically possible in a value function.

The second advantage is that we can easily formulate criteria with various dependencies. Suppose that the car has another attribute of whether the bumper is chrome or painted in the color of the car. The decision maker might not like a black bumper, since it is easily scratched, so he would prefer the black color only if the bumper is chrome. In a value function, this means that color and the type of bumper are no longer preferentially independent, so that we need to elicit a combined value function for all combinations. In a constraint-based model, P_2 can simply become a soft constraint on both color and bumper type, without having to affect the other options.

Some of the constraints could be inferred from a personal profile or using methods such as collaborative filtering, de-

mographical analysis, default logic, and case-based reasoning (Ha & Haddawy 1998). For example, knowing that a traveler is booking a business trip, we can infer that his most important criterion is to be at his destination on time. However, most constraints will be specific to the particular scenario, and need to be explicitly stated by the user. In this paper, we focus on user stated preferences.

Allowing users to state arbitrary criteria as soft constraints is a significant user interface challenge. We have found it sufficient to decide a certain number of useful parameterized criteria when designing the system and make these accessible through the user interface. While this choice limits the preferences that can be stated by a particular user, the framework itself places no limit on what criteria can be used for preferences. Depending on how much interface complexity a user is willing to accommodate, it is possible to provide a very rich vocabulary for formulating them. For example, a travel planning system we have built allows users to specify unary and binary constraints on any combination of attributes visible in the display. This flexibility is an important advantage over database-based systems where the set of possible preferences is strongly restricted by the schema of the underlying database.

For each criterion, the system designer devises a parameterized function (criterion function) that maps each value combination of the involved attributes to a numerical value. This function is chosen to fit the majority of users. For example, in an apartment search application, price differences are counted linearly, while preferences for location are counted with a step function that assigns 0 utility to non-preferred values and utilities in increasing steps of 100 to the more preferred ones. Each preference has a weight that the user is able to change. The function is inaccurate because it is standardized for all users. However, we have shown in earlier work that such inaccuracies can be compensated by generating a set rather than a single optimal solution (Faltings *et al.* 2004; Faltings, Torrens, & Pu 2004). Other researchers have investigated methods for obtaining constraints from examples, for example (O'Sullivan, Freuder, & O'Connell 2001).

Preference Revision through Tradeoffs

We now consider how to support preference revision through tradeoffs in a decision framework. Tradeoff is an important aspect of preference construction as it is here that decision makers refine the details of their preferences into an accurate model. Furthermore, refining a preference model through tradeoffs focusses the user's effort on those parts of the preference model that are actually decisive for picking the final outcome, thus playing an important role in reducing the elicitation effort.

¿From observing user behaviors, we have identified the following 3 tradeoff strategies used by human decision makers:

- value tradeoff: a user explicitly changes the preference values of a set of attributes; for example, the user can state a value tradeoff to prefer a flight with a more favorite airline while compromising 2 hours on the total flight time.

	Price	Surface	Location	Bus
1	800	25	Center	12min
2	600	24	Renens	15min
3	900	30	Morges	8min
4	800	35	Renens	2min

Table 1: Several apartment choices.

- utility tradeoff: a user changes the weights of a set of preferences so that those attributes with higher weight values take precedent when values are assigned; for example, when airline preference is becomes more important than the total flight time, then the system will try to satisfy the airline preference before satisfying the flight time preference.
- outcome tradeoff: a user performs either value or utility tradeoff after viewing a set of outcomes; for example, seeing a particularly good airline in one of the outcomes makes him either want to state an explicit preference for that airline, increase the weight of airline preference, or both.

Supporting these tradeoff strategies requires particular agility of the preference model that cannot be effectively supported by rigid elicitation procedures. As an example to illustrate how a constraint-based model supports the three types of tradeoff, consider the 4 choices for apartments shown in Table 1.

A user may state the following 3 initial criteria:

1. (weight = 1): I am willing to pay up to 700.:
   $u1 = 700 - price$
2. (weight = 20): Surface has to be at least 30.:
   $u2 = Surface - 30$
3. (weight = 100): I work in the center, so I prefer Center over the suburbs Morges over Renens:
   $u3 = (caselocation : \begin{cases} Center : 2 \\ Morges : 1 \\ Renens : 0 \end{cases})$

We assume that preferences are combined by forming a weighted sum, so the initial ranking assigned to the 4 outcomes would be:

$$\begin{aligned} u(1) &= -100 + 20 * -5 + 100 * 2 = 0 \\ u(2) &= 100 + 20 * -6 + 0 = -20 \\ u(3) &= -200 + 0 + 100 * 1 = -100 \\ u(4) &= -100 + 20 * 5 + 0 = 0 \end{aligned}$$

Outcomes 1 and 4 are preferred over 2 and 2 is preferred over 3. However, none of the outcomes provides a positive utility, so the decision maker is not likely to pick any of them as a final solution.

Tradeoffs are required whenever none of the outcomes achieves a sufficiently high ranking according to the preference model. This then requires a user to change the preference model so that an acceptable solution can be found. In the constraint-based decision framework, the three different tradeoff strategies described in the introduction can be modeled as follows:

1. Value tradeoff: the user changes one of the soft constraints in his preference model, increasing or decreasing the utility of a certain value combination in comparison with the others.
2. Utility tradeoff: the user changes the weight that a particular soft constraint is given in the combined ranking.
3. Outcome tradeoff: the user adds additional soft constraints that increase the relative utility of certain choices to make them acceptable.

In the example, none of the outcomes is ranked with a utility greater than his threshold of 0. Now the decision maker can engage in the three kinds of tradeoff:

1. an example of a value tradeoff would be to reverse the preferences for location after realizing that Morges offers the best school for the decision maker's children and thus should be most preferred. This can be done by adding a soft constraint with a weight of 200 that gives a value of 1 to Morges and none to the other locations. Since both soft constraints are added up, this has the effect of making Morges the most preferred location, but requires no revision of the earlier soft constraints. Now, the third solution has a value of 100 and is thus acceptable.
2. an example of a utility tradeoff would be to change the weight of surface preference from 20 to 30. Now choice 4 has a positive utility of 50 and could be chosen. This can be done by modifying the weight of the preference in the weighted combination function.
3. an example of an outcome tradeoff would be to notice that choice 4 is in fact very close to public transport, and add the following additional preference with weight 10: $u4 =$ (if Bus $<$ 3min: 10, otherwise 0). Now choice 4 has a positive utility of 100 and is acceptable.

Note that in each case, the required modification of the ranking is achieved by adding a preference or changing its weight in the combination model. In the soft constraint model, these kinds of modifications are easy to carry out and provide an agile modeling process. Such agility would be very hard to achieve in models based, for example, on classical preference elicitation, where the impact of answering a particular query is hard to evaluate.

Note furthermore that each of these three tradeoff types requires that the decision maker has a good understanding of the available outcomes and how they are affected by preferences. This understanding is provided, at least to some degree, by the example-critiquing framework: by showing examples that are already close to the user's desires, it increases his understanding of the possibilities in the outcome space that is relevant to his desires. In further work, we have also shown techniques for generating additional examples that are likely to stimulate users to express preferences that have not been stated so far (Faltings *et al.* 2004). These methods are based on a probabilistic analysis of what examples are likely to become optimal should the user discover new criteria.

Conversely, the constraint-based decision framework provides the agile and easily modifiable preference model that is required to implement example critiquing. The synergy between the two models leads to powerful practical systems.

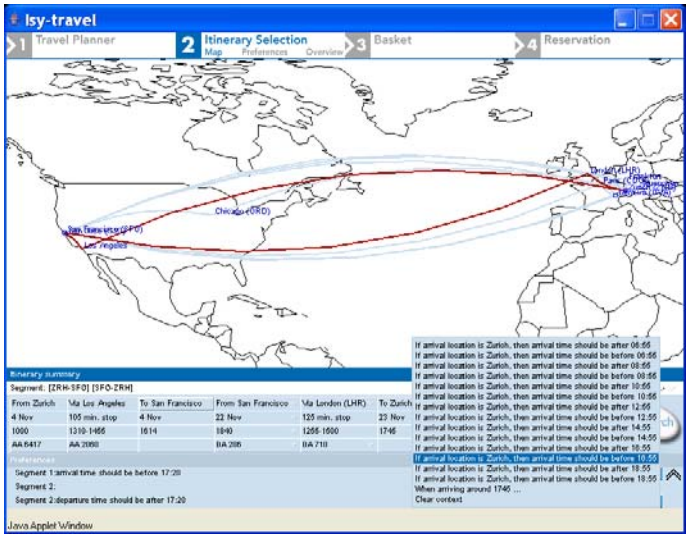

Figure 2: *ISY-travel allows users to add preferences by posting soft constraints on any attribute or attribute combination in the display. Preferences in the current model are shown in the preference display at the bottom, and can be given different weights or deleted.*

Example: ISY-Travel

We have used example critiquing with constraint-based preferences models in ISY-travel, a tool for travel planning that was commercialized by Iconomic Systems and later i:FAO (Pu & Faltings 2000; Torrens, Faltings, & Pu 2002; Pu & Faltings 2002). In ISY-travel, the user starts by giving dates and destinations of travel. The tool then gathers all available airline schedules that may be relevant to the trip, and generates 30 examples of solutions that are good according to the current preference model. The preference model is initially preset with a number of common-sense preferences, such as short travel time, few connections, low price, etc. Seeing the examples, the user incrementally builds his preference model by adding preferences as shown in Figure 2.

Preferences can be added on any attribute or pair of attributes and in any order. Preferences on pairs of attributes arise when a user conditions a preference for one attribute on another one, for example one can select a different preferred departure time for each possible departure airport. Preferences can also be removed or given lower or higher weight by operations in the preference panel. When the preference model has been sufficiently modified, the user can ask the system to recompute again the 30 best solutions according to these preferences.

When there are too many preferences, it can happen that there is no longer a single solution that satisfies them all. In this case, the system shows solutions that satisfy the preferences to the largest degree possible. For example, in Figure 3, the user has posted constraints on the departure and

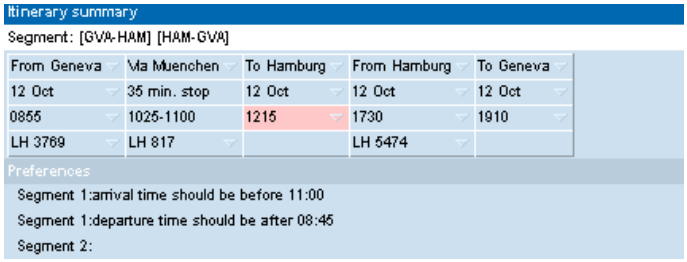

Figure 3: *When it is not possible to satisfy all preferences completely, ISY-travel looks for solutions that satisfy as many of them as possible and acknowledges the attributes that violate preferences in red.*

arrival times that cannot both be satisfied. Thus, the system proposes solutions that satisfy only one of the preferences, and acknowledges the violation by showing the attribute in question on a red background.

User studies have shown that the agile preference model used in ISY-travel has led users to find travel itineraries that were much closer to their wishes with less effort than with traditional tools such as travelocity.

Conclusion

A number of researchers from both behavioral and qualitative decision theory have pointed out the advantage of eliciting user preferences incrementally. We have shown how to go further by allowing users to actively *construct* a preference model in an example-critiquing interaction.

Such interaction is enabled by agile preference models that allow users to add, retract and modify any criterion they choose in any order. We have shown how this agility is provided by preference models based on soft constraints. Such models provide the flexibility required to deal with the highly dynamic nature of human preferences in such interactions.

References

Allen, J. F.; Schubert, L. K.; Ferguson, G. M.; Heeman, P. A.; Hwang, C. H.; Kato, T.; Light, M. N.; Martin, N. G.; Miller, B. W.; Poesio, M.; ; and Traum, D. R. 1994. The TRAINS project: A case study in building a conversational planning agent. In *TRAINS Technical Note 94-3*.

Boutilier, C.; Patrascu, R.; Poupart, P.; and Schuurmans, D. 2004. Regret-based utility elicitation in constraint-based decision problems. In *Working Paper*.

Burke, R. D.; Hammond, K. J.; and Young, B. C. 1997. The FindMe approach to assisted browsing. *IEEE Expert* 12(4):32–40.

Carenini, G., and Poole, D. 2002. Constructed preferences and value-focused thinking: Implications for ai research on preference elicitation. In *Proceedings of the AAAI workshop on Preferences in AI and CP: Symbolic Approaches*. Edmonton, Canada: AAAI.

Faltings, B.; Pu, P.; Torrens, M.; and Viappiani, P. 2004. Designing example-critiquing interaction. In *International Conference on Intelligent User Interfaces*, 22–29. Island of Madeira (Portugal): ACM.

Faltings, B.; Torrens, M.; and Pu, P. 2004. Solution generation with qualitative models of preferences. *Computational Intelligence* 20(2):246–263.

Ha, V., and Haddawy, P. 1997. Problem-focused incremental elicitation of multi-attribute utility models. In *Proceedings of the Thirteenth Conference on Uncertainty in Artificial Intelligence*, 215–222.

Ha, V., and Haddawy, P. 1998. Toward case-based preference elicitation: Similarity measures on preference structures. In *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*, 193–201. San Francisco: Morgan Kaufmann Publishers.

Horvitz, E. 1999. Principles of mixed-initiative user interfaces. In *CHI '99: Proceedings of the SIGCHI conference on Human factors in computing systems*, 159–166. ACM Press.

Keeney, R., and Raiffa, H. 1976. *Decisions with Multiple Objectives: Preferences and Value Tradeoffs*. John Wiley and Sons, New York.

Keeney, R. 1992. *Value-Focused Thinking. A Path to Creative Decision Making*. Cambridge: Harvard University Press.

Linden, G.; Hanks, S.; and Lesh, N. 1997. Interactive assessment of user preference models: The automated travel assistant. In *Proceedings, User Modeling '97*.

O'Sullivan, B.; Freuder, E.; and O'Connell, S. 2001. Interactive constraint acquisition. In *roceedings of Workshop on User-Interaction in Constraint Processing at the CP-2001*.

Payne, J.; Bettman, J.; and Johnson, E. 1993. *The Adaptive Decision Maker*. Cambridge University Press.

Pu, P., and Faltings, B. 2000. Enriching buyers' experiences: the smartclient approach. In *SIGCHI conference on Human factors in computing systems*, 289–296. ACM Press New York, NY, USA.

Pu, P., and Faltings, B. 2002. Personalized navigation of heterogeneous product spaces using smartclient. In *International Conference on Intelligent User Interfaces*. ACM Press.

Ruttkay, Z. 1994. Fuzzy constraint satisfaction. In *Proceedings of the 3rd IEEE International Conference on Fuzzy Systems*, 1263–1268.

Schiex, T.; Fargier, H.; and Verfaillie, G. 1995. Valued constraint satisfaction problems: Hard and easy problems. In *IJCAI'95: Proceedings International Joint Conference on Artificial Intelligence*, 631–639.

Shearin, S., and Lieberman, H. 2001. Intelligent profiling by example. In *Proceedings of the Conference on Intelligent User Interfaces*, 145–151. ACM Press New York, NY, USA.

Torrens, M.; Faltings, B.; and Pu, P. 2002. Smartclients: Constraint satisfaction as a paradigm for scaleable intelligent information systems. *CONSTRAINTS: an Internation Journal. Kluwer Academic Publishers* (7):49–69.