

ence vectors. This leads to an efficient update scheme that maintains an incremental utility estimate that is relatively stable, and provides a efficient test for violation of the linear utility assumption.

Third, the system uses a more expressive language than many systems performing incremental utility elicitation. The utility model is a linear combination of attributes that are themselves open to modification and creation by the user. For example, while preferring low cost alternatives in a travel planning system, the user can easily incorporate a discount on certain airlines in the computation of flight cost, essentially linking the two attributes. Tools provided in Constable (Blythe *et al.* 2001) allow the attributes to be modified or new attributes to be defined through an English paraphrase. The system will also suggest modifications to the attributes when the assumption of a linearly additive utility is violated.

In the next section we discuss existing systems for interactive utility elicitation and describe the flight planning domain, used for examples in this paper. Next we describe the VEIL tool that integrates visual exploration and IUE and show an example of a user's interaction with the system. The next section describes the utility vector update scheme used in VEIL, followed by empirical results with different update algorithms and different measures of convergence. Next we show how users can edit the attributes used in the utility vector and how changes are suggested by the tool. Finally we discuss the relation to existing approaches and describe possible future work.

Incremental utility elicitation

In the general multi-attribute decision problem (Keeney & Raiffa 1993) we have a set O of outcomes and a set S of strategies, each of which has an associated probability distribution over the outcomes. Our aim is to find a strategy that is preferred by the user based on her preference structure over outcomes. We assume the user is rational, so the preferences can be expressed by a utility function $u^*(\cdot)$ that maps O to the real numbers, such that outcome o_1 is preferred to o_2 if and only if $u^*(o_1) > u^*(o_2)$. The outcomes are described by a set of attributes A where each $a_i \in A$ is a function $a_i : O \rightarrow \text{domain}(a_i)$. The utility $u^*(\cdot)$ is assumed to be a function of the attributes. We do not have access to the user's true utility $u^*(\cdot)$ but instead create a model $u'(\cdot)$ based on information from the user. We refer to $u'(o)$ as the *estimated utility* of o . In the rest of this paper, we assume a correspondence between strategies and outcomes, so the user task is to choose an outcome. We refer to the outcomes as alternatives.

Choosing a round-trip flight between two locations is an example of this problem. The set O of alternatives is the set of available flights, described by attributes like cost, time of departure and arrival. Many tools exist on the web that can retrieve flights and allow the user to browse them through a fixed scheme, for example for browsing by price, airline, or time of departure or arrival. However these sites can easily generate thousands of flights for a query and a fixed browsing scheme can be cumbersome if it does not match the user's requirements (Pu & Faltings 2000).

Visual exploration of alternatives

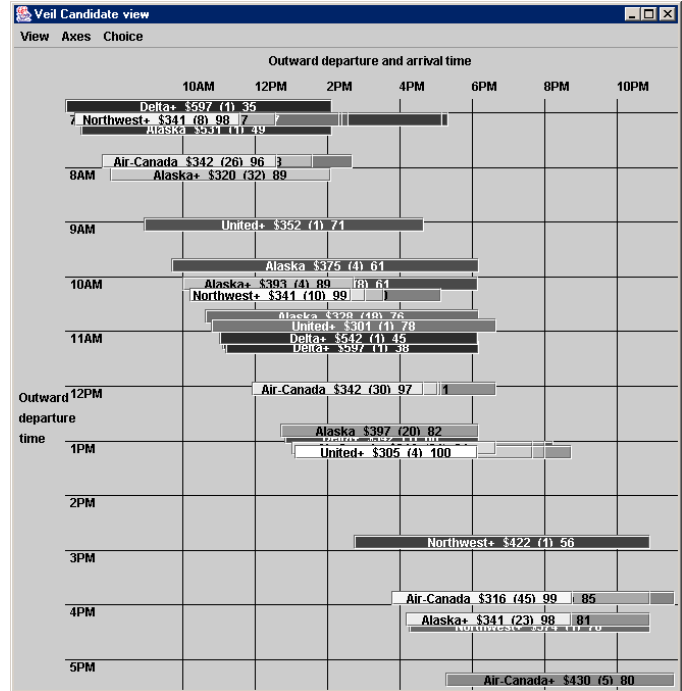


Figure 1: A projection of the alternative flights by outbound departure time, with width showing outbound duration and with grey-scale coloring according to the initial estimated utility.

In the VEIL system, users interact with a two-dimensional projection of the alternatives corresponding to two attributes used for the x and y coordinates, and optionally others depicted with shape, width, height and a textual label. For example, Figure 1 shows a projection of alternatives for a round-trip flight from Los Angeles to Victoria, British Columbia. In this projection, both x and y coordinates are based on the outbound departure time, while the width of each bar shows the duration of the outbound flight. Users can change the projection at any time by choosing the attributes to use for each axis, and the projection is automatically scaled — this layout was chosen because no rectangle is likely to completely obscure another. The flight data were retrieved from the web using wrappers, and represent 500 alternatives. However, since many round-trip flights share the same outbound leg, only 91 distinct alternatives are visible and each visible rectangle accounts for a group of 5 flights on average. The label on each rectangle first shows the airline and cost of the represented flight that has the maximum estimated utility for the group. Next to the cost is shown the number of separate alternatives in the group, and finally the maximum estimated utility, re-scaled to range from 0 to 100. Users can select a rectangle to bring it to the front of the display or to see more information about that choice in a separate window.

This display broadly follows the Star field approach (Ahlberg & Shneiderman 1994), which is a popular tech-

nique for displaying large sets of multi-attribute data. We chose to organize alternatives spatially by two attributes because the attributes in travel planning carry intuitive meaning. Other display methods use techniques such as multi-dimensional scaling that project the alternatives onto two dimensions in a more flexible way (Marks *et al.* 1997). While these approaches can place items more efficiently and avoid occlusion, it is harder to draw conclusions about the attributes of items from their positions.

Two features of the layout are chosen to help the user find high-utility alternatives easily. First, the grey-scale shading of each rectangle is chosen to represent its rank in the ordering based on utility, with the best alternative in white and the worst in black. Second, because the layout of alternatives is directly proportional to two attributes, the rectangles for alternatives may overlap as in Figure 1. When this happens, the alternative with higher estimated utility is placed on top.

Figure 1 shows a default utility that minimizes cost and the outbound and return flight durations, with equal weight after the attributes are normalized. The display provides the user with summary information despite the large number of alternatives. For instance, there are good flights leaving at regular intervals through the day until about 4pm. The preferred option according to the default utility is the United flight leaving at 1pm.

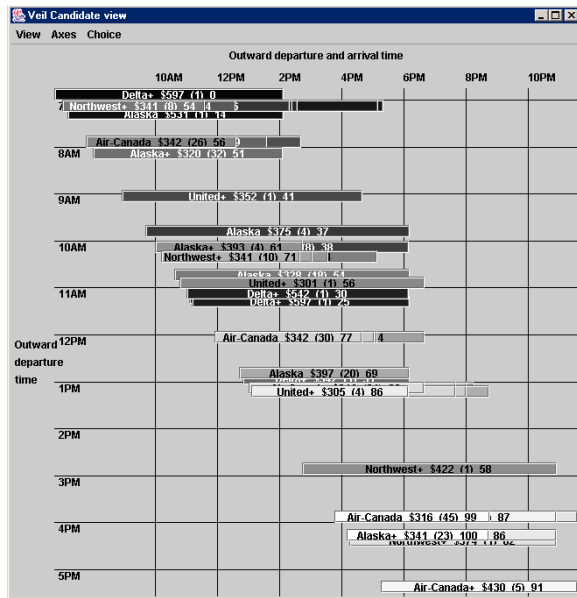


Figure 2: Choices with grey-scale shading according to the adjusted utility

For the purpose of finding an optimal solution, the value of this tool is limited by how well the utility estimate matches the user's true utility. The user can update the utility estimate directly, through a window where each attribute and its weight is shown and can be changed. The utility can also be updated indirectly by stating a direct preference on the alternatives displayed on the screen. In this example, the user prefers to depart as late as possible, and expresses this

as a preference for the Air Canada flight departing at 1pm over the flight departing at 12pm. Figure 2 shows the resulting utility estimate that has been adjusted to take this preference into account. Preferred flights are now mostly found in the lower right corner of the screen, and the new preferred alternative is the Alaska flight leaving after 4pm. The next section describes how the utility function is updated based on the preference.

Using estimated utilities to both order and highlight good flights significantly helps users in the task of searching for good flights. As an illustration, we inspected a version of the display in Figure 1 but with uniform colors and a random ordering of overlapping rectangles. Since some flights were obscured, we found either a Northwest flight leaving at 4pm or a United flight leaving at 1pm. The option suggested by the tool is significantly better than these.

Interaction style

Can the graphical system support the same kind of interactions that decision makers are used to? We analyzed an email conversation with a travel planner to pick a flight, fragments of which are shown below, and compared the activities it contained with the types of interaction supported by our tool. The activities can all be classified as seeking or providing information, either about available alternatives or about the utility function.

The first passage suggests an option, explains why it cannot be improved on one dimension but suggests an alternative that improves on this feature by making other compromises. This suggestion also elicits utility information.

Here is a tentative itinerary. Leaving Victoria on the 22nd, can't get much later due to the time change and flight times from Seattle - DC. There is a red-eye that leaves at 10.30pm out of Seattle if you are interested in leaving later?

VEIL suggests options with high estimated utility but does not offer explanatory information. The reply both provides utility information about specific domain attributes and seeks information about alternatives.

Thanks. Is there anything later leaving on the 20th? [outbound leg]. The flight from Victoria to Dulles is ok, I don't want a red-eye.

VEIL allows preferences between options to change utility weights, providing this information as we showed with the example above. The information sought could be found from a projection of the alternatives.

Updating the utility model

As well as directly changing weights an alternatives, the user can update the utility by expressing a direct preference on alternatives in the display, as described above. This is sometimes preferable because the user does not need an explicit understanding of the relative weights of different features in the utility function. It can also provide more information because in addition to indicating attributes whose weights should change, it also gives a lower bound on the amount of change required in order to accommodate the new preference.

VEIL uses a linear estimate for utility, so

$$u'(o) = \sum_{i=1}^n w_i u_i(o)$$

where the utility function has n components $u_i()$, and w_i is the weight attached to the i th component. For example, one component might be the cost of a flight, another might be its duration and a third might encode a preference over the airline used. The components are normalized so that each ranges from 0 to 1 in value over all outcomes under consideration.

For convenience we identify the outcome o with the vector of component utility functions $\mathbf{o} = (u_i(o), 1 \leq i \leq n)$, and write the summation as a dot product, so $u'(o) = \mathbf{w}' \cdot \mathbf{o}$, the dot product of the weight and utility vectors. When a new pairwise preference is introduced to the system, say $\mathbf{a} > \mathbf{b}$ for alternatives \mathbf{a} and \mathbf{b} , this is interpreted as a linear constraint on \mathbf{w}' :

$$\mathbf{w}' \cdot \mathbf{a} > \mathbf{w}' \cdot \mathbf{b}$$

so

$$\mathbf{w}' \cdot (\mathbf{a} - \mathbf{b}) > 0$$

We refer to $\mathbf{a} - \mathbf{b}$ as the *preference vector*. Each preference vector therefore induces a linear constraint on the weight vector \mathbf{w}' , a half-space in which \mathbf{w}' must lie. There is a solution for \mathbf{w}' if and only if the polytope defined by the intersection of the half-spaces orthogonal to each preference vector is non-empty. Any vector lying in the polytope will constitute a solution. However it is computationally expensive to enumerate the faces of the convex hull in more than a few dimensions, so instead we solve a linear programming (LP) problem whose constraints include the preference vectors:

$$\begin{aligned} &\text{Maximize } \mathbf{c} \cdot \mathbf{w} \\ &\text{Subject to: } \mathbf{A} \mathbf{w} > \mathbf{0}, \\ &\quad -1 \leq w_i \leq 1 \text{ for } 1 \leq i \leq n \end{aligned}$$

where the vector \mathbf{w} has n dimensions and the preference matrix \mathbf{A} has rows corresponding to the preference vectors. The final conditions on the w_i ensure that the solution is bounded. If this LP problem is infeasible, then the original assumption of linearity in the utility model is violated. Otherwise, any solution will correspond to a utility function that respects the user's preferences, regardless of the choice of the objective function, specified by the vector \mathbf{c} . We adopt the solution \mathbf{w}' as the new utility estimate.

There are several choices for \mathbf{c} that lead to interesting properties for \mathbf{w}' . If we choose $\mathbf{c} = \mathbf{w}'_o$, the previous utility estimate, the LP makes the new estimate as close as possible to the old one, leading to a conservative update algorithm. If we choose $\mathbf{c} = \sum_i a_i$, the sum of the preference vectors, the new estimate will be as close as possible to an average of the user's stated preferences. In practice we solve for a small number of different values of \mathbf{c} in order to sample the space of feasible utilities.

Our utility update algorithm adds a new row to the preference matrix \mathbf{A} and solves the new LP problem whenever the user adds a preference. We skip this step if the previous utility is feasible and the objective function would choose

it. In VEIL, the LP problems are solved using the Simplex algorithm. Although this is known to have exponential worst-case behavior, the algorithm is often competitive with the polynomial-time interior-point method in practice (Spielman & Teng 2001). We have not found the running time of the LP solver to be a problem, but a potential improvement is to maintain a set of extremal preference vectors, so that the rank of \mathbf{A} is as low as possible while still representing the same polytope.

In most domains we would like the tool to begin with a default utility that captures common preferences, for instance to minimize duration and cost of both outbound and return parts of the roundtrip flight. These are captured by preference vectors that are unioned with the set of user-specified preference vectors. The size of the convex space defined by the vectors can be adjusted to control the latitude that the user has to deviate from the default utility.

Choosing the features to update

The basic update scheme above might produce unexpected utility weights when only a small number of preferences is given in a high-dimensional space. In addition, decision makers can often point to a reason why one alternative is preferred over another, in terms of a subset of the features of the alternatives, although they may find it hard to describe an explicit utility function. To exploit of this information, the user can also nominate a subset of the features whose weights are to be altered when a preference is given. We first attempt to update the utility estimate while modifying only these weights, by solving the linear programming problem created from the original by setting the remaining variables to their values in the current utility estimate. If there is no feasible solution for this more constrained LP we relax the conditions and seek a solution with the original set of non-zero weights.

The nominated subset of attributes is chosen when the user expresses the preference in the alternatives display, through a menu that shows the attributes of the alternatives that could be used as components of the utility function. Attributes that have the same value for the two alternatives are not selectable and are shown greyed out in the menu. Using a subset of features allows a more natural interaction with the decision maker, including expressions like 'I prefer this flight because it arrives later'.

Experiments

VEIL includes two different ways that a decision maker can modify the utility estimate: either by altering the weights directly or by expressing pairwise preferences on the alternatives that are displayed in the visualization. The decision maker must have a good idea of how to interpret her preferences in terms of global weights in order to use the direct approach effectively. Since updates based on pairwise preferences do not require this knowledge, one can ask how much more interaction is generally required for the tool to converge on a utility estimate that well represents the decision maker's preferences. Perhaps surprisingly, our experiments indicate that, in some cases, little extra interaction is required.

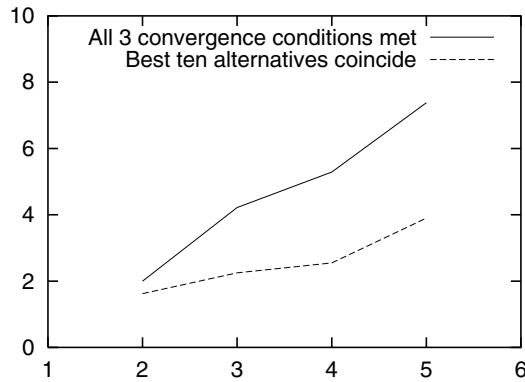


Figure 3: Rates of convergence for utility update based on pairwise preferences, as a function of the number of linear components in the 'true' utility.

Before describing experiments that support this, we discuss notions of convergence for utility estimates. The strictest form of convergence is to ask that the estimated utility and the decision maker's true utility function be identical up to a scale factor (which will not affect preferences). However, a less precise match will generally be adequate for the decision-making task and might require less work from the user. One might measure the utility loss, in terms of the decision maker's true utility, of accepting the suggested alternative based on the estimated utility. However in a visual tool like VEIL, the user will also inspect a number of alternatives apart from the best under the estimated utility.

Therefore we include two other measures of convergence. First, the Kendall rank correlation coefficient τ measures how close are the rankings on the alternatives induced by the two utility functions on the set of available alternatives (Noether 1986). This is the difference of the proportion of pairs of alternatives in the same order in each ranking and the proportion of pairs in the opposite order. The value of τ ranges from 1 for identical rankings to -1 for precisely opposite rankings. For the second measure, we take into account the limited cognitive effort that the decision maker is likely to spend examining alternatives. We assume that the user will check only the best ten items according to the estimated utility, and test whether this coincides with the best ten items in the true utility measure, although perhaps in a different order. There are clearly many variations of these measures that might be appropriate in certain situations, but these are sufficient to compare the rates of convergence of the two utility update schemes in VEIL.

For our experiments, we assumed that the decision maker's true utility is a linear combination of between 2 and 5 features. The features are known to the system, but not their correct weights. The tool begins with a uniform weighting of features and selects a pair of alternatives such that the preference in the true utility is the reverse of the preference in the estimated utility. This preference is used to update the estimated utility using the linear programming technique described above. We considered three convergence

measures: the Kendall rank correlation > 0.95 , the utility loss zero (*i.e.* a maximally preferred item is selected by the estimated utility), and finally the condition that the best ten alternatives in the true utility are also the best ten under the utility estimate.

For each value of n we ran at least 30 trials with randomly generated utility functions on 500 real flights generated in response to a travel request. The upper line in Figure 3 shows the average number of pairwise preferences that have to be specified for all three convergence measures to be met. It can be seen that the average number of preferences required is one or two more than the number of features involved for this range of values. A direct update to the utility function would require up to $n - 1$ weights to be set, so the amount of interaction required for VEIL to estimate the utility function based on pairwise preferences is comparable. The lower line on the graph shows the average number of preferences required for convergence according to the 'best ten' measure.

In the experiment, knowledge of the true utility was used to select a pair of alternatives whose utilities do not reflect the true preference, a task we expect the user to perform in real use. Global knowledge of the true weights is not necessary for the algorithm to converge, however in order to achieve the convergence rates shown in Figure 3, the pairs were selected so that their utility difference was above a threshold. When random pairs of alternatives were selected, convergence was considerably slower. It seems plausible that users will provide information about preferences that are clearly different from those of the current estimate rather than preferences with only a slight difference, but user experiments are needed to test this hypothesis.

Editing the features for expressivity

The tool makes the assumption that the utility can be modelled as the weighted sum of attribute values. When this assumption is violated, the violation is detected in our utility update scheme when the linear program has no feasible solutions. In this case the tool can suggest ways to change the attributes that are used in the linear sum. First we describe how the features themselves can be changed, with some examples of typical changes.

The attributes that form the linear components of the utility estimate in VEIL are represented as declarative functions written in the EXPECT language (Blythe *et al.* 2001). This system includes tools that allow users to form expressions based on the domain features that include iteration and conditional tests, as well as arithmetic and set theoretic operations. A flight cost utility component can be modified to model a 15% discount on United, for example, or a departure time component can model 'departure time close to 9am or close to 5pm'.

Rather than store the preference vectors directly, VEIL stores the pairs of outcomes for which the user expressed a preference. Thus when the component utility features are changed, the preference vectors corresponding to the outcome pairs can easily be recomputed. The set of preferences nominated by the user is likely to be fairly small, making this approach feasible.

Recovering from a violated linearity assumption

We note that a piecewise linear utility function can be defined to capture at least the ordering of a finite set of alternatives in any utility. The boundaries of the piecewise linear utilities can be represented within each feature computation, preserving the summation form of the overall utility model. When a new preference vector leads to the linearity assumption being violated, we find a minimal set of preference vectors that violate the assumption (ie, are linearly dependent). This must include the most recently added preference vector. Next we look at the features nominated in the preferences, if any, and suggest modifying one of these. For each one, a break-point is suggested with alternative weights on either side, which are chosen so that the corresponding LP is feasible.

Although we treat the violation of the linearity assumption as evidence that the component utility features are inadequate to represent the user's utility, users often enter inconsistent preferences in practice for a variety of reasons. While indicating a minimal set of inconsistent preferences and showing how the utility functions may be adjusted to accommodate them, VEIL may also allow the user to modify the preferences in case they were in error or the user's utility function is changing.

Discussion

VEIL is a decision-making aid that integrates incremental utility elicitation with visual exploration of alternatives. VEIL provides an information-rich environment for the decision maker, while using an incremental utility estimate to help guide the search for a good alternative. The utility estimate can be updated based on preferences that are expressed directly in the exploration tool, or by modifying the weights. The resulting system can make a suggestion at any time, and the user can terminate the session at any time.

VEIL can detect when the linearity assumption is violated and offer suggestions to recover. The recovery strategy may be either to enrich the utility measure or to modify the user's preferences. In this respect, the combination of a visual environment with immediate feedback on changes in the utility model may allow users more quickly to catch mistaken preferences, or those with unexpected consequences.

Although we gave examples and described experiments in the domain of air travel planning, the principles and techniques used are domain independent. VEIL is also being applied to a special operations forces planning domain, but we omit details for reasons of space.

Ha and Haddawy (1997) motivate incremental utility elicitation and describe a system that constructs queries for the user. They propose a heuristic based on rank correlation for choosing the question to ask. Chajewska et al. (2000) describe an approach to incremental utility elicitation that makes use of a prior probability distribution on possible utility models. At each step, their algorithm constructs a question for the user with optimal value of information, stopping when the expected utility loss is below a threshold. Our approach does not assume a prior distribution on utilities, and follows a principle of providing more information and

more control for the decision maker. Thus, we allow users to supply preference information rather than answer queries constructed by the system. Although the system can be terminated at any time, we use stopping criteria that consider the ordering of some or all of the alternatives to show empirical rates of convergence. Hanks et al. (1997) describe the Automated Travel Assistant that presents a shortlist of examples to the user and allows direct updates to the utility weights. Like us, Pu and Faltings (2000) use a visual approach to help a user choose a flight, but they do not make use of utility estimates.

Work in mixed-initiative problem solving also emphasizes shared control between the user and the computer program and has typically been applied to planning or scheduling problems (Ferguson, Allen, & Miller 1996; Anderson et al. 2000; Myers 1996). The problem addressed by VEIL is less complex than these, but because of its mixed-initiative strategy it may be an appropriate component in more sophisticated mixed-initiative systems, helping to present results and giving users an opportunity to influence the problem-solving behaviour of the system.

The application of VEIL within more complicated planning and scheduling tasks is something we intend to study in the future. Other areas to explore are how the existence of a population of users or tasks can be exploited within this framework. Intuitively, the utility functions of a decision maker for two different flights are likely to be quite similar, perhaps depending on the destination and purpose of travel. Chajewska et al. (1998) have used clustering techniques that might be applicable in VEIL through the objective functions or constraints used in its linear program formulations. There are also many directions to explore with both the utility update and the visualization portions of VEIL. Our initial work shows this to be a promising approach in an increasingly important research area.

Acknowledgments

Discussions with Yolanda Gil, Jihie Kim and Fred Bobbit were very helpful, as were the comments of the anonymous reviewers. I gratefully acknowledge support from DARPA contracts F30602-00-2-0513 as part of the Active Templates program, and N66001-00-C-8018, with subcontract number 34-000-145 to SRI International, as part of the Rapid Knowledge Formation program.

References

- Ahlberg, C., and Shneiderman, B. 1994. Visual information seeking: Tight coupling of dynamic query filters with starfield displays. In *Human Factors in Computing Systems. Conference Proceedings CHI'94*, 313–317.
- Anderson, D.; Anderson, E.; Lesh, N.; Marks, J.; Mirtich, B.; Ratajczak, D.; and Ryall, K. 2000. Human-guided simple search. In *Proc. Seventeenth National Conference on Artificial Intelligence*. AAAI Press.
- Blythe, J.; Kim, J.; Ramachandran, S.; and Gil, Y. 2001. An integrated environment for knowledge acquisition. In *Proc. Conference on Intelligent User Interfaces*.

Chajewska, U.; Getoor, L.; Norman, J.; and Shahar, Y. 1998. Utility elicitation as a classification problem. In *Proc. Fourteenth Conference on Uncertainty in Artificial Intelligence*, 79–88. Madison, Wisconsin: Morgan Kaufmann.

Chajewska, U.; Koller, D.; and Parr, R. 2000. Making rational decisions using adaptive utility elicitation. In *Proc. Seventeenth National Conference on Artificial Intelligence*, 363–369. AAAI Press.

Ferguson, G.; Allen, J.; and Miller, B. 1996. Trains-95: Towards a mixed-initiative planning assistant. In Drabble, B., ed., *Proc. Third International Conference on Artificial Intelligence Planning Systems*. University of Edinburgh: AAAI Press.

Ha, V., and Haddawy, P. 1997. Problem-focused incremental elicitation of multi-attribute utility models. In Besnard, P., and Hanks, S., eds., *Proc. Thirteenth Conference on Uncertainty in Artificial Intelligence*, 215–222. Providence, Rhode Island: Morgan Kaufmann.

Keeney, R., and Raiffa, H. 1993. *Decisions With Multiple Objectives*. The Pitt Building, Trumpington Street, Cambridge, UK: Cambridge University Press.

Linden, G.; Hanks, S.; ; and Lesh, N. 1997. Interactive assessment of user preference models: The automated travel assistant. Sixth International Conference on User Modelling.

Marks, J.; Andalman, B.; Beardsley, P.; Freeman, W.; Gibson, S.; Hodgins, J.; and T.Kang. 1997. Design galleries: A general approach to setting parameters for computer graphics and animation. In *Proc. of SIGGRAPH*, 389–400.

Myers, K. 1996. Strategic advice for hierarchical planners. In *Proceedings of the International Conference on Knowledge Representation*.

Noether, G. E. 1986. Why kendall tau? In *Best of Teaching Statistics*. Teaching Statistics. available at <http://science.ntu.ac.uk/rsscse/TS/bts/noether/text.html>.

Pu, P., and Faltings, B. 2000. Enriching buyers' experiences: the smartclient approach. In *ACM CHI Conference on Human Factors in Computing Systems*, 289–296.

Spielman, D., and Teng, S.-H. 2001. Smoothed analysis: Why the simplex algorithm usually takes polynomial time. In *The Thirty-Third Annual ACM Symposium on Theory of Computing*, 296–305.

von Neumann, J., and Morgenstern, O. 1967. *Theory of Games and Economic Behaviour*. Princeton University Press, 3rd edition.