The Lookahead Principle for Preference Elicitation: Experimental Results

Paolo Viappiani[1], Boi Faltings[1], and Pearl Pu[2]

[1] Artificial Intelligence Laboratory (LIA)
[2] Human Computer Interaction Group(HCI)
Ecole Polytechnique Fédérale de Lausanne (EPFL)
1015 Lausanne, Switzerland
{paolo.viappiani, boi.faltings, pearl.pu}@epfl.ch

Abstract. Preference-based search is the problem of finding an item that matches best with a user's preferences. User studies show that example-based tools for preference-based search can achieve significantly higher accuracy when they are complemented with suggestions chosen to inform users about the available choices.

We discuss the problem of eliciting preferences in example-based tools and present the lookahead principle for generating suggestions. We compare two different implementations of this principle and we analyze logs of real user interactions to evaluate them.

1 Introduction

People increasingly rely on web applications to search products in online catalogs. It is common to let the user express preferences on attributes and then let a database system find the most preferred item according to these preferences. We call this task *preference-based search*.

The most common search facility is based on a form that is directly mapped to a database query and returns a list of the most suitable options. The user has the option to return to the initial page and change his preferences and then carry out a new search. This is the case for example when searching for flights on the most popular travel web sites[1][2]. Such tools are only as good as the query the user formulates. A study [3] has shown that among the users of such sites only 18% are satisfied with their final choice.

Database researchers have studied query systems that evaluate predicates with a continuous degree of validity and allow partial matches, as in fuzzy sql (`FSQL`) [2] and `Preference SQL` [8].

A key issue for preference-based search systems is how to acquire or learn preferences from the user.

One way to obtain the user model is to elicit it by a set of questions. However, it has been shown that this can lead to significant inaccuracies in the user model,

[1] `http://www.travelocity.com/`
[2] `http://www.expedia.com`

H. Legind Larsen et al. (Eds.): FQAS 2006, LNAI 4027, pp. 378–389, 2006.

because users may not be able to give the correct answer at the time that they are asked by the elicitor [14]. In most cases, users do not know exactly what they are looking for: they might consider different trade-offs or they might even have conflicting desires about the features the item should have. Psychological studies [11] have shown that people construct their preferences while learning about the available products. Therefore preference-based search should also help users in formulating accurate preferences.

An alternative that often results in models of higher quality is to let users volunteer their preferences. For example, in [10] it has been shown that in a collaborative filtering system, letting users themselves propose items they want to rate yields better results than a strategy where the items are chosen to optimally elicit the preference model.

An interesting technique for letting users volunteer their preferences is an interaction where the system shows proposed options and lets users express their preferences as *critiques* stimulated by these examples. This technique is called *example* or *candidate* critiquing, and has been explored by several authors [5, 9, 18, 17].

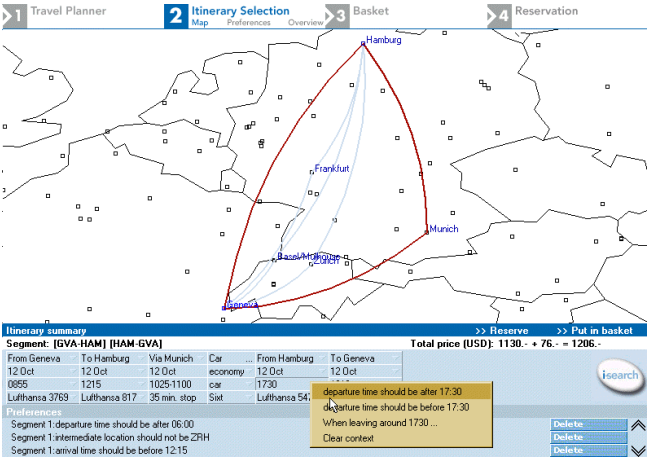

Fig. 1. Isy-travel is an example-critiquing tool for planning business-trips. Here the user is planning a one-day trip from Geneva to Hamburg. The preference model is shown at the bottom, and the user can state additional preferences by clicking on features of the shown example.

Figure 1 shows Isy-travel, a commercial tool for business travelers [12]. Here, the user is shown examples of options that fit the current preference model well. The idea is that an example either is the most preferred one, or there is some aspect in which it can be improved. Thus, on any of the examples, any attribute can be selected as a basis for critiquing. For instance, if the arrival time is too late, then this can be critiqued. The critique then becomes an additional preference in the model.

The advantage of such a system in the elicitation of preferences is that examples help users reason about their own preferences, revise them if they are inconsistent, have an idea of which preferences can be satisfied and make trade-off decisions.

Example critiquing achieves higher decision accuracy when the displayed options are complemented with suggestions chosen to inform users about available choices [16]. The cognitive effort is comparable to simple interfaces such as a ranked list [15].

In this paper, we discuss the role of suggestions in stimulating preference expression in form of critiques. We present two alternative implementations of the *lookahead* principle on which the generation of suggestions is based and show the complexity of the computation. Finally we make an evaluation on some real user interaction logs.

2 Incremental Preference Model Acquisition

In example-critiquing, preferences are stated as reactions to displayed options. We can classify such critiques according to the context in which the statement of a preference takes place:

negative reaction none of the options shown satisfy that preference
positive reaction an option is shown that satisfies (partially or completely) the preference

For instance, if the tool shows the user examples that all arrive at London Stansted airport, and she requests to land in Heathrow, that critique would be a *negative reaction*. If the system indeed showed one flight landing in Heathrow, by stating that preference she would be *positively* reacting to the shown examples.

An option that partially satisfies the preference can also cause preference expression. If the user sees examples of flights landing in Stansted and Gatwick, by seeing the possibility of landing in Gatwick, considered better than Stansted but worse than Heathrow, she is stimulated to state a preference about the landing airports.

This simple distinction is quite fundamental in understanding the cognitive process of constructing preferences and designing the elicitation process. If certain preferences are missing from the current model of the user, the system provides examples that do not satisfy those unknown preferences. If the user is aware of all of her preferences, she can realize the necessity to state them to the system by posting what we have called a *negative reaction* critique. However our intuition is that this is not always the case, because the user might not know all the available options. Moreover, stating a preference costs some user effort (in our prototype, 2 selections and 2 clicks) and she would make that effort only if she perceives this as beneficial.

To use a metaphor, the process of example-critiquing is hill-climbing: the user states preferences as long as she perceives it as bringing to a better solution. However, the process might end in a local optimum; a situation in which the

user can no longer see potential improvement. For example, a user looking for a notebook computer might start looking for a low price, and thus find that all models weigh about 3 kg. Since all of the presented models have about the same weight, he or she might never bother to look for lighter models. This influence of current examples prevents the user from refocussing the search in another direction; this is known as the *anchoring effect* [19].

For these reasons the displayed set consists of two parts:

- a **candidate** set of options that are optimal for the preference model, and
- a **suggested** set of options that are chosen to optimally stimulate the expression of preferences.

2.1 Suggestions: Diversity and Lookahead Principle

The importance of the diversity of the example shown was recognized by Linden, S. Hanks and N. Lesh [9] who explicitly generated examples that showed the extreme values of certain attributes, called *extreme examples*. However, an extreme example might often be an unreasonable choice: it could be a cheap flight that leaves in the early morning, a student accommodation where the student has to work for the family, an apartment extremely far from the city. Moreover, in problems with many attributes, there will be too many extreme or diverse examples to choose from, while we have to limit the display of examples to few of them.

We assume that the user is minimizing her own effort and will add preferences to the model only when she can expect them to have an impact on the solutions. This is the case when:

- she can see several options that differ in a possible preference, and
- these options are relevant, i.e. they could be reasonable choices, and
- these options are not already optimal, so a new preference is required to make them optimal.

In all other cases, stating an additional preference is likely to be irrelevant. When all options would lead to the same evaluation, or when the preference only has an effect on options that would not be eligible anyway, stating it would only be wasted effort. This leads us to the following *lookahead* principle as a basis for suggestion strategies:

> Suggestions should not be optimal under the current preference model, but should provide a high likelihood of optimality when an additional preference is added.

We stress that this is a heuristic principle based on assumptions about human behavior that we cannot formally prove. However in the last section we will provide empirical evidence of the correctness of this principle.

3 Theoretical Model

3.1 Modeling Items and Preferences

We assume that items are modeled by a fixed set of m attributes that each take values in associated domains. Domains can be *enumerated*, consisting of a set of discrete elements, or *numeric*. In this paper, we consider preferences on individual attributes, i.e. we do not consider conditional preferences. A preference r is an order relation of the values of an attribute a.

For a practical preference-based search tool, it is convenient to express preferences in a concise way. We consider total orders (each pair is comparable) and express them by a numerical cost function c, $d_k \rightarrow \Re^+$, that maps a domain value d_k of an attribute a_k to a real number. A preference always applies to the same attribute a_k; we use the notation $c_i(o)$ to express the cost that the function assigns to the value of option o for that attribute. Whenever o_1 is preferred to o_2 according to preference i, the first will have lower cost (for preference i) than the second: $c_i(o_1) < c_i(o_2)$.

An overall ranking of options can be obtained by combining the cost functions for all stated preferences. Some researchers [5] have proposed the use of machine learning algorithms for finding the best aggregate function for a particular user. In our systems, we combine them using a weighted sum, which corresponds well to standard multi-attribute utility theory [6]. Thus, if $\mathcal{R}_c = \{c_1, .., c_s\}$ is the set of the cost functions of all preferences that the user has stated, we compute the cost $C(o) = \sum_{c_i \in \mathcal{R}_c} w_i \cdot c_i(o)$. Option o_1 is preferred over option o_2 whenever it has a lower cost, i.e. $C(o_1) < C(o_2)$.

The user states preferences in a qualitative way (for example "the price should be less than 500 dollars"). We map these qualitative statements into parameterized functions that are standardized to fit average users. These are chosen with respect to the application domain.

Preference modeling for example-critiquing is discussed in more detail in [13]. In [7], the authors propose a similar model of preferences that allows prioritizing constraints and skyline queries.

3.2 Dominance Relation and Pareto Optimality

We model preferences by standardized functions that correctly reflect the preference order of individual attribute values but may be numerically inaccurate. When generating suggestions we would like to use a model that is not sensitive to this numerical error. We thus use the qualitative combination concepts of *dominance* and *Pareto optimality*:

An option o is **(Pareto) dominated by** another option $\bar{o}$ (equivalently we say that $\bar{o}$ dominates o) if

- $\bar{o}$ is not worse than o according to all preferences in the preference model: $\forall c_i \in R : c_i(\bar{o}) \leq c_i(o)$
- $\bar{o}$ is strictly better than o for at least one preference: $\exists c_j \in R : c_j(\bar{o}) < c_j(o)$

An option o is **Pareto-optimal** if it is not Pareto-dominated by any other option. The dominance relation is a partial order of the options that we will denote with the $\succ$ operator; Pareto-optimal options can also be seen as the set of maximal options with respect to the dominance relation.

The **dominating set** is the set of options that dominates a particular option.

3.3 Model-Based Suggestion Strategy

In [16] we proposed different strategies that use the concept of pareto optimality to implement the lookahead principle stated in the introduction: suggestions should not be optimal yet, but have a high likelihood of becoming optimal when an additional preference is added. We call them *model-based* suggestion strategies because they specifically choose examples to stimulate the expression of additional preferences based on the current preference *model*.

In our applications, users initially state only a subset R of their true preference model $\overline{R}$. When a preference is added, dominated options with respect to R can become Pareto-optimal. The lookahead principle can be formulated as follows: an ideal suggestion is an option that is Pareto-optimal with respect to the full preference model $\overline{R}$, but is dominated in R, the partial preference model.

The model based suggestions try to guess the chance that a (dominated) option has to become Pareto-optimal. This can happen when a new preference is added to the model and the option is strictly better than any dominating option with respect to this new preference.

Supposing that o_d dominates o, we use a heuristic estimation of the probability that a hidden preference on attribute a_i makes o better than o_d according to that preference, hence escaping the dominance relation. Such a heuristic considers the difference between the attribute values: the higher this difference, the more likely a new preference will make the option preferred. The reasoning is illustrated in Figure 2. The chances that a new preference will treat o_1 and o_2 differently depends on the difference between their values. Assuming that the shape of such a cost function is a step function with a sharp increase from 0 to 1 and the reference point falls at any point with equal probability, then the chance of breaking the dominance is directly proportional to this difference.

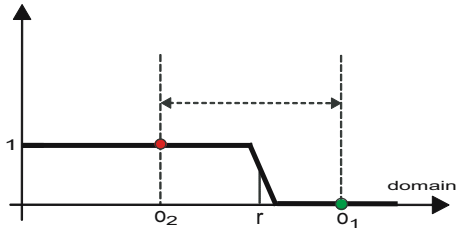

Fig. 2. For an ordered attribute, a new preference will prefer o_1 over o_2 if the reference value r falls between the values of the attribute, and the preference is of the right polarity

For attributes with enumerated domains, it is sufficient to check if the attributes takes different values. If so, there will be equal chances that one is preferred over the other. If the values are the same, the dominance relation cannot be broken by a preference on this attribute.

3.4 Utility Dominance

Other forms of dominance can be defined as extensions of Pareto-dominance such that if o Pareto-dominates o' then o also dominates o'. In particular, we might use the total ordering established by the combination function defined in the preference modeling formalism, such as a weighted sum. We call this *utility-domination*, and the utility-optimal option is the most preferred one.

An option can become utility-optimal only if it is strictly better than all options that currently utility-dominate it, although this is not a sufficient condition. The utility dominance method consists of checking the probability of breaking utility dominance. The advantage is that the dominating set is easily computed by simply checking the cost: once we have the ranking for the current preferences, the utility-dominating set will be composed by all the options with a higher ranking.

4 Algorithms

In this section we analyze the algorithms to compute the candidates and suggestion examples and their complexity.

4.1 Generation of Candidates

Preference-based search looks for the option that best satisfies a preference model, given an aggregate cost function that merge the individual preferences. The best options can be found by sorting the database items according to their cost. This is known as the *top-k query* [4]. The set of options retrieved $\{o_1, .., o_k\}$ is such that $C(o_1) \leq C(o_2) \leq .. \leq C(o_k)$ and for any other option $\bar{o}$ in the database $C(\bar{o}) \geq C(o_k)$.

While the trivial approach would compute the score of each option in the database, the *Fagin* algorithm [4] can do this with complexity $O(N^{(m-1)/m}k^{1/m})$ where m is the number of attributes and k the number of candidates we want to generate. This algorithm can be applied to a different aggregate function as long as it satisfies some properties (monotonicity, strictness). Even faster optimizations have been recently proposed [1].

4.2 Generation of Suggestions

The model-based strategy requires the analysis of the dominance relation as a preliminary phase, so that it is possible to associate a given option to its set of dominators (the options that dominate it). Pareto dominance or utility dominance can be used.

Algorithm 1. Model-based suggestions(int n)
δ heuristics based on the normalized differences

for all *option* $\in OPTIONS$ **do**
  $p(option) = 0$
  for all $a_i \in A_u = \{$attributes with no preferences$\}$ **do**
    $p(option, a_i) = 1$ //contribution for attribute a_i
    for all $o_d \in O_D = \{o_d \in O : o_d \succ option\}$ **do**
      //we iterate over the set of options that dominate o
      $\bar{\delta} \leftarrow \delta_i(o, o_d)$
      $p(option, a_i) = \text{update}(p(option, a_i), \bar{\delta})$
  $p(option) = 1 - \prod_i (1 - p(option, a_i))$
suggList $\leftarrow$ order options according p
return first n options in *suggList*;

We stress that the computation of the suggestions requires more effort than the Pareto or skyline queries (that do not return the dominating sets).

The algorithm for model-based suggestions is presented in Algorithm 1. *Update* is responsible for updating the value for the estimation of the probability $p(o, a_i)$ of becoming Pareto-optimal given that the missing preference is on a_i; its precise definition depends on the particular assumptions on the possible preferences.

In the *probabilistic* strategy, the update multiplies the current value by $w_i * \delta_i(o, o_d)$, where $\delta_i(o, o_d)$ is the heuristic estimation presented in the previous section (based on the normalized distance between the attribute values) calculated for attribute a_i and w_i is a weight representing the probability that there is a preference on that attribute. Intuitively, the more dominating options there are, the more $p(o, a_i)$ decreases.

In another model-based strategy, the *attribute* strategy, we assume that the preferences that the user can state are only of the kind `LessThan` or `GreaterThan` (the user cannot express preferences for a value in the middle). Therefore, for each attribute we check whether the current option has a value that is either smaller or bigger than any value of the dominating options: only in this case a preference can break all the dominance relations simultaneously. In this strategy, `update` will take the minimum of the absolute values of the $\bar{\delta}$, and returns 0 if they are of different signs.

Complexity. The main issue is that we do not simply need to find Pareto-optimal options but also need to know all the options that dominate a given one. Thus, most of the optimizations used by skyline queries cannot be used.

The analysis of the Pareto dominance relation makes a series of pairwise checks between the options of the catalog. Each one evaluates two options, say o_1 and o_2, to determine whether o_1 dominates o_2, o_2 dominates o_1, they are equally preferred for all the preferences, or they are not comparable (i.e. there is no dominance in either direction). This is done by considering iteratively each of the preferences and comparing o_1 and o_2 for at most m comparisons (as soon as two preferences give opposite order of o_1 and o_2, they are not comparable),

where m is the number of preferences, so the complexity is $O(m)$. In the worst case we have to make $n(n-1)/2$ checks. So the complexity of the complete dominance analysis is $O(n^2m)$.

Our model based strategies have complexity $O(nmd)$, where m is the number of attributes and d is the number of dominating options. It is difficult to calculate an average value for d in function of n, because it depends on the data and correlation between attribute values. While generally the set of dominating options is much smaller than the set of options, in the worst case they can be a linear fraction of n. Sorting the options according to the resulting probability (to select the best n) costs $nlogn$ in term of complexity.

The overall complexity is $O(n^2)$: while this complexity was not a problem for our prototype, we expect it to be more problematic as the item collections grow. Approximations will be necessary for large databases.

If utility dominance is considered, the dominating set of an option is the set of all options with lower cost. However this simplification of the preliminary phase does not improve the overall complexity of the generation of suggestions. In fact we have to make, for each option, a comparison to its utility-dominators. These are 1 for the first option in the rank, 2 for the second, and n for the last. Again, we have a total complexity of $O(n^2)$.

5 Evaluation

In this section, we evaluate the lookahead principle through real user interactions. Previously [16] we conducted user studies to evaluate the decision accuracy of example critiquing with and without suggestions on a prototype for student accommodation search called $FlatFinder$. After picking their choices with different versions of the tool, users were asked to determine their best choice by carefully examining the entire database. The decision of the tool was deemed accurate whenever this choice agreed with the tool. We found that with the aid of suggestions, users state more preferences and, more importantly, achieve a much higher decision accuracy (up to 70%).

We then investigated more closely how people state preferences, using the logs from this user test and from more recent experiments. A total of 100 interaction logs were considered. 40 of these were interactions of users using the interface without suggestions (showing 6 candidate optimal examples) and the rest were interactions of users using the interface with suggestions (3 candidates and 3 suggestions shown at each cycle), calculated according to the two semantics (Pareto and utility dominance).

We expected to find an empirical confirmation of our lookahead principle for generating suggestions.

First, we counted the frequency of the different types of critiquing described before. As shown in Table 1, in most cases (79%) a preference is stated as a positive reaction to one of the displayed options. This supports our intuition: users are likely to state preferences when they see examples that show options that differ in a possible preference.

Table 1. In the majority of cases a preference is stated when the user sees an example that satisfies it (positive reaction to the displayed options)

Critiques	Frequency
Positive reactions	0.79
Negative reactions	0.21

Table 2. Summary of the user experiment comparing the interface with and without suggestions (average per user). The initial preferences are the ones stated in the initial cycle of the interaction, before having seen any example, while the final preferences are those in the last interaction cycle. The number of critiques on new attributes are a particular case of preference revision. Decision accuracy was evaluated by asking the subjects to carefully examine the entire database of offers to determine their target option, that was compared with the choice made with the search tool.

	Interface without suggestions	Interface with suggestions
interaction time (min.)	8:09	7:39
number of initial preferences	3.03	3.30
critiques (preference revisions)	5.96	6.49
critiques on new attributes	2.65	3.45
number of final preferences	4.90	5.70
decision accuracy	45%	75%

Table 3. Evaluation of the implementations of the lookahead strategy for each type of the interface: no suggestions, model-based suggestion implemented with Pareto dominance relation, model-based suggestions with utility dominance. The table shows the fraction of positive critiques that are either Pareto or utilitarian critiques. In most cases a critique is stated when there is a displayed example that becomes Pareto-optimal because of the addition of a preference.

	Fraction of positive critiques for each interface			
Critiques	No sugg.	Suggestions (Pareto)	Suggestions (Utility)	Overall
Pareto critiques	47%	60%	42%	49%
Utilitarian critiques	36%	33%	35%	35%

Surprisingly, there was no significant difference in the fraction of positive critiques between the users of the interface with suggestions and those of the interface without suggestions. However as shown in Table 2 the first group made more critiques on new attributes (3.45 vs 2.65) on average, and achieved higher decision accuracy (75% against 45%) meaning that the preferences acquired with the suggestions are more accurate.

Then, we looked more closely at the positive critiques to support the second part of our intuition, that the user states a preference when she sees options that can be a reasonable choice and a new preference is required to make them optimal. We checked how many times the user stated a preference that made one

of the displayed options optimal, considering the two possible definition of dominance relation: Pareto dominance and Utility dominance. We call **Pareto** the critiques in which a Pareto-dominated option becomes Pareto-optimal; **Utilitarian** the critiques in which a Utility dominated option becomes Utility optimal.

In Table 3 we show the results for different kind of interactions: interface without suggestions, interface with suggestions computed with pareto-dominance and suggestions generated with utility-dominance. Overall, nearly half (49%) of the positive critiques are stated as Pareto critiques; the fraction increases up to 60% when suggestions are generated according to this principle. In all the circumstances, the frequency of Pareto critiques dominates that of utilitarian critiques. Therefore, Pareto optimality seems to be a reasonable way to implement the lookahead principle.

6 Conclusions

Preference-based search is a ubiquitous problem on the web. Tools based on examples can achieve higher decision accuracy than the traditional form filling approach. User studies show the importance of suggestions to make the users aware of possible choices and stimulate the preference expression.

We presented the lookahead principle that identifies good suggestions as items that have high likelihood of becoming optimal when other possible preferences are considered. By examining user behavior we found empirical evidence that suggestions are an important means to stimulate the user to refine the query. Most of the times (79%) users state preferences when they see examples that are perceived as improvement.

We discussed two possible implementations of the principle, one based on Pareto optimality and one based on a Utility ranking. The evaluation shows that the first seems to better represent the cognitive process of the user as it explains up to 60% of the cases.

References

1. W.-T. Balke and U. Güntzer. Multi-objective query processing for database systems. In M. A. Nascimento, M. T. Özsu, D. Kossmann, R. J. Miller, J. A. Blakeley, and K. B. Schiefer, editors, *VLDB*, pages 936–947. Morgan Kaufmann, 2004.
2. B. P. Buckles and F. E. Petry. Fuzzy databases in the new era. In *SAC*, pages 497–502, 1995.
3. M. S. D. W. Equity. Transportation e-commerce and the task of fulfilment, 2000.
4. R. Fagin. Fuzzy queries in multimedia database systems. In *PODS '98: Proceedings of the seventeenth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems*, pages 1–10, New York, NY, USA, 1998. ACM Press.
5. S.-w. H. Hwanjo Yu and K. C.-C. Chang. Rankfp: A framework for supporting rank formulation and processing. In *ICDE 2005*, pages 514–515, 2005.
6. R. L. Keeney and H. Raiffa. *Decisions with Multiple Objectives: Preferences and Value Tradeoffs.* John Wiley and Sons, New York, 1976.

7. W. Kiesling. Foundations of preferences in database systems. In *VLDB 2002*, pages 311–322, 2002.
8. W. Kießling and G. Köstler. Preference sql - design, implementation, experiences. In *VLDB*, pages 990–1001, 2002.
9. G. Linden, S. Hanks, and N. Lesh. Interactive assessment of user preference models: The automated travel assistant. In *Proceedings, User Modeling '97*, 1997.
10. S. M. McNee, S. K. Lam, J. A. Konstan, and J. Riedl. Interfaces for eliciting new user preferences in recommender systems. In P. Brusilovsky, A. T. Corbett, and F. de Rosis, editors, *User Modeling 2003*, LNCS 2702, pages 178–187. Springer, 2003.
11. J. Payne, J. Bettman, and E. Johnson. *The Adaptive Decision Maker*. Cambridge University Press, 1993.
12. P. Pu and B. Faltings. Enriching buyers' experiences: the smartclient approach. In *SIGCHI conference on Human factors in computing systems*, pages 289–296. ACM Press New York, NY, USA, 2000.
13. P. Pu and B. Faltings. Decision tradeoff using example-critiquing and constraint programming. *Constraints: An International Journal*, 9(4), 2004.
14. P. Pu, B. Faltings, and M. Torrens. Effective interaction principles for online product search environments. In *Proceedings of the 3rd ACM/IEEE International Conference on Web Intelligence*. IEEE Press, September 2004.
15. P. Pu and P. Kumar. Evaluating example-based search tools. In *ACM Conference on Electronic Commerce (EC'04)*, 2004.
16. P. Pu, P. Viappiani, and B. Faltings. Increasing user decision accuracy using suggestions. In *CHI*, page to appear, April 2006.
17. H. Shimazu. Expertclerk: Navigating shoppers buying process with the combination of asking and proposing. In *Proceedings of the 17 International Joint Conference on Artificial Intelligence (IJCAI'01)*, volume 2, pages 1443–1448, 2001.
18. B. Smyth and L. McGinty. The power of suggestion. In *IJCAI*, pages 127–132, 2003.
19. A. Tversky. Judgement under uncertainity: Heuristics and biases, 1974.