Designing Example-critiquing Interaction

Boi Faltings
Artificial Intelligence Laboratory (LIA)
Swiss Federal Institute of Technology(EPFL)
1015 Lausanne, Switzerland
boi.faltings@epfl.ch

Pearl Pu
Human Computer Interaction Group
Swiss Federal Institute of Technology(EPFL)
1015 Lausanne, Switzerland
pearl.pu@epfl.ch

Marc Torrens, Paolo Viappiani
LIA, EPFL
1015 Lausanne, Switzerland
marc.torrens@epfl.ch
paolo.viappiani@epfl.ch

ABSTRACT

In many practical scenarios, users are faced with the problem of choosing the most preferred outcome from a large set of possibilities. As people are unable to sift through them manually, decisions support systems are often used to automatically find the optimal solution. A crucial requirement for such a system is to have an accurate model of the user's preferences.

Studies have shown that people are usually unable to accurately state their preferences up front, but are greatly helped by seeing examples of actual solutions. Thus, several researchers have proposed preference elicitation strategies based on *example critiquing.* The essential design question in example critiquing is what examples to show users in order to best help them locate their most preferred solution.

In this paper, we analyze this question based on two requirements. The first is that it must stimulate the user to express further preferences by showing the range of alternatives available. The second is that the examples that are shown must contain the solution that the user would consider optimal if the currently expressed preference model was complete so that he select it as a final solution.

Keywords

Decision Support Tools, Example Critiquing

1. INTRODUCTION

Many real-world applications require people to select a most preferred item from a set of choices. For example, in e-commerce an electronic catalog system might provide access to millions of products. The user has to navigate the catalog to find the most preferred one, which we call the *target.* Decision theory ([8]) provides algorithms which guarantee to find this target solution given an accurate numerical preference model.

Studies have shown that people are usually unable to accurately state their preferences up front, but are greatly helped by seeing examples of actual solutions. Thus, several researchers have proposed preference elicitation strategies based on *example critiquing.* Example critiquing is a process where the system shows a set of candidate solutions generated on the basis of the user's current preference model, and the user reacts by either picking one as his choice or modifying his preference model.

Some of the earliest successful example critiquing systems are FindMe ([2]) and ATA ([10]). FindMe has been implemented for a variety of domains, such as finding movies or restaurants, and has been the one of the earliest systems to use example critiquing in a decision aid tool. [10] describes a preference elicitation method for a travel planning system, ATA. The ATA system (automated travel assistant) uses a constraint solver to obtain a set of optimal solutions. Five of them are shown to the user, three optimal ones in addition to two extreme solutions (least expensive and shortest flying time). A candidate critiquing agent (CCA) constantly observes user's modification to the expressed preference, and refines the elicited model in order to improve solution accuracy. A similar approach was taken by Smart Client ([15, 12, 16]) which performs travel planning with a wider range of criteria and shows a larger set of 30 possible solutions in different visualizations. It has been developed into a commercial system, *reality* ([17]). Apt Decision ([13]) uses learning techniques to synthesize a user's preference model by observing their critique of example apartment features. Its main objective is profiling and predicting what a user wants in searching for an apartment.

As a concrete example of an example-critiquing interaction, Figure 1 shows an screenshot of the SmartClient tool used for planning a trip from San Francisco to Geneva. It shows a range of 30 example solutions that can be visually compared according to to various attributes such as departure and arrival times, travel time, transfer aiports and time, airlines, aircraft types, etc. Users can critique examples by posting preferences on attributes or combinations of attributes. Figure 1 shows an example where the user adds a preference on the transit time. This preference translates into a penalty function that assigns a penalty to solutions proportional to the degree that the preference is violated. The set of penalty functions obtained in this way is the model of the user's preferences maintained by SmartClient and used to generate the displayed examples. Every change in the preference model leads to an instantenous update of the displayed solutions. At any time, the user can choose to

IUI'04, January 13–16, 2004, Madeira, Funchal, Portugal.
Copyright 2004 ACM 1-58113-815-6/04/0001 ...$5.00.

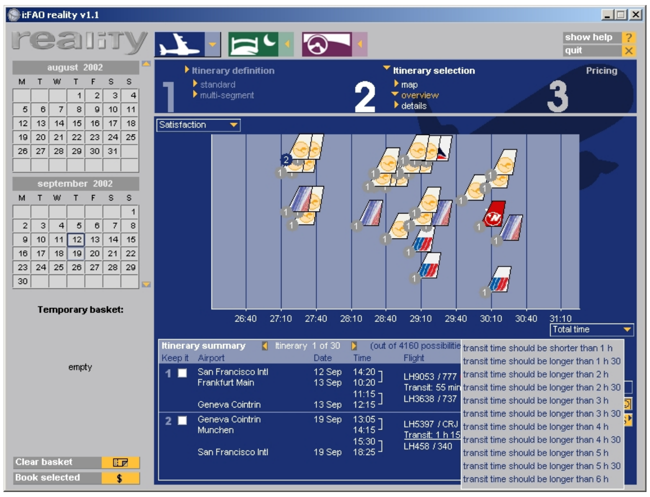

Figure 1: *An example-critiquing interface based on the SmartClient architecture.*

select one of the examples as the final solution.

A crucial design question in example critiquing is what examples to show to users in order to best help them locate their most preferred solution. This is driven by two essential considerations: (i) the examples must motivate the user to correctly state his preferences, and (ii) when the user has completely stated his preferences, the most preferred solution must be among those displayed by the system so the user can choose it.

More precisely, we can assume that users are best motivated to state hidden preferences when they see both an example that satisfies and one that violates the preference. Decision support tools can easily mislead users: for example, the user may be shopping for a notebook computer and the first preference he states might be price. A list of examples that simply optimizes his preference model would lead him to conclude that all notebook computers weigh about 3Kg and have a 14-inch screen, and probably he would never ask for lighter or smaller computers. Thus, he would be greatly helped if the system also showed at least one example of a very light, but more expensive notebook.

Another issue is that users usually are not able to formulate a numerically precise model of their preferences. While mathematical decision theory ([8, 9]) provides normative methods for picking the optimal solution, the effort required for eliciting the parameters of a numerical preference function ([6, 7]) is often too high. Simplified models such as the PC selection tool of IBM described in [14], where users can adjust the weights of different criteria and interactively see how different PC models rank according to these weights, do not actually guarantee that a correct preference model is elicited either, as users are generally not able to state their preferences as weights.

Instead, most example critiquing systems characterize the user's preference *qualitatively* as the specific combination of criteria that apply, without eliciting numerical weights. Such qualitative decision theory has recently become the subject of increased interest in the research community ([4, 1]). In example critiquing, we can compensate for the inaccuracies of such a model by displaying a set of solutions and letting the user choose the optimal one within this set. The important design question is then how many and what solutions to show to guarantee that the user is actually able to choose his most preferred one. In this paper, we show how under certain assumptions, we can guarantee that the most prefereed solution will be shown.

2. ASSUMPTIONS AND DEFINITIONS

As a basis for our analysis, we present a formal model of the example-critiquing process that closely mirrors the SmartClient technology mentioned earlier, but also applies to a large extent to other example-critiquing systems.

The goal of example-critiquing interaction is to select a *solution* from a large set of possibilities. We assume that solutions are characterized by a fixed (but possibly very large) set of *attributes*, and that users formulate *preferences* in the form of penalty functions formulated on the attributes. We thus define:

DEFINITION 1. *Every solution s is characterized by a finite set of n attributes $A = \{a_1, ..., a_n\}$. Every attribute can take values from a fixed domain $\{d_1, ..., d_n\}$. Attributes may express a composition, such as the difference, of other attributes.*

A preference $p_i(a_k)$ *is a penalty function $d_k \rightarrow \Re$ from an attribute a_k to a number that gives the penalty of that attribute value to the user. We assume that the smallest values are the most preferred ones. Considering that a preference p_i always applies to the same attribute a_j, we simplify the notation and write $p_i(s)$ for $p_i(a_j(s))$.*

We assume that the true preferences of the user are given by a set of penalty functions $P^* = \{p_1^*, .., p_d^*\}$. We call the best solution for the user's true preference model P^* the user's *target solution* s_t.

However, since the user is not aware of this numerically precise model, he expresses his preferences *qualitatively* by choosing penalty functions and parameters as afforded by the interface. For example, SmartClient provides a predefined numerical penalty function for each attribute. The user can activate this penalty function by clicking on the attribute. While many penalty functions include parameters chosen by the user - for example, a preference on arrival time has to include the most preferred time - the numerical shape of the function itself is fixed and identical for all users.

Thus, in our analysis we assume that the example-critiquing system has a fixed set of parameterized penalty functions that are used to build a numerically precise user preference model. This model is used to generate the examples displayed to the user.

Here is an example of what attributes and actual penalty functions might look like:

EXAMPLE 1. *Consider the example of planning a trip consisting of two flights f_1 and f_2 that connect at a transfer airport.*

- *A solution could be characterized by the following attributes:*
  $a_0 =$ `departure_time` (f_1)
  $a_1 =$ `arrival_time` (f_2)
  $a_2 =$ `departure_time` (f_2) - `arrival_time` (f_1) *: transit time*
  $a_3 =$ `transfer_airport` (f_1, f_2)
  $a_4 =$ `arrival_time` (f_2) - `departure_time` (f_1) *: travel time*

- *The system could provide the following penalty functions for specifying preferences, where time and length are user-specified parameters:*
  $latest(time, a_i) = max(0, a_i - time)$
  $earliest(time, a_i) = max(0, time - a_i)$
  $min - duration(length, a_i) = max(0, length - a_i)$
  $max - duration(length, a_i) = max(0, a_i - length)$
  $rule - out(value, a_i) = 1$ *if* $a_i = value$, *0 otherwise*

- *Suppose that the user states a preference for arriving by 18:00, for having at least 2 hours transit time and for not changing planes in London. These qualitative statements would then be translated into the following numerically precise penalty functions:*
  $latest(18:00, a_1) = max(0, a_1 - 18:00)$
  $min - duration(2:00, a_2) = max(0, 2:00 - a_2)$
  $rule{-}out(London, a_3) = 1$ *if* $a_3 = London$, *0 otherwise*

The system thus approximates the user's preferences $P^* = \{p_1^*, .., p_d^*\}$ by a set $P = \{p_1, .., p_d\}$ of standard penalty functions. While the standard functions p_i may be quite different from the true ones p_i^*, we assume that the total order that a penalty function p_i by itself imposes on the set of solutions is identical to that of p_i^*. Formally, for any two solutions s and s', we assume:

$$p_i(s) < p_i(s') \leftrightarrow p_i^*(s) < p_i^*(s')$$

It follows from this assumption that the preference model correctly represents the user's preferences with respect to *dominance* and *Pareto-optimality*, defined as follows:

Definition 2. *A solution s is* dominated *with respect to P if and only if there is another solution s' such that for all $p_i \in P$, $p_i(s) \geq p_i(s')$ and at least one $p_j \in P$, $p_j(s) > p_i(s')$. We write $s \prec s'$.*

A solution s_p is Pareto-optimal *if and only if it is not dominated.*

In Figure 2, the Pareto-optimal set is $\{1, 3, 4, 6\}$, as solution 7 is dominated by 4 and 6, 5 is dominated by 3 and 4, and 2 and 8 are dominated by 1.

In some analyses, we assume furthermore that the user follows a particular model for combining preferences:

- in the *utilitarian* model, preferences are combined in a quasilinear cost function $C(s) = \sum_{p_i^* \in P^*} p_i^*(s)$. Solution s_1 is preferred over solution s_2 whenever it has a lower cost, i.e. $C(s_1) < C(s_2)$.

- in the *egalitarian* model, user preferences are combined by considering the maximum penalty, i.e. the function $F(s) = max_{p_i^* \in P^*} p_i^*(s)$. Again, solution s_1 is preferred over solution s_2 whenever it has a lower maximum penalty, i.e. $F(s_1) < F(s_2)$.

In both of these preference models, we assume that the standardized preference functions are normalized for average users so that the inaccuracy of p_i with respect to p_i^* is bounded by ϵ:

$$(1 - \epsilon)p_i \leq p_i^* \leq (1 + \epsilon)p_i$$

This inaccuracy reflects different degrees of importance that different users attach to violations of a preference. Note that we assume that each normalized penalty function still correctly represents the qualitative ordering of solutions when all other criteria are equal.

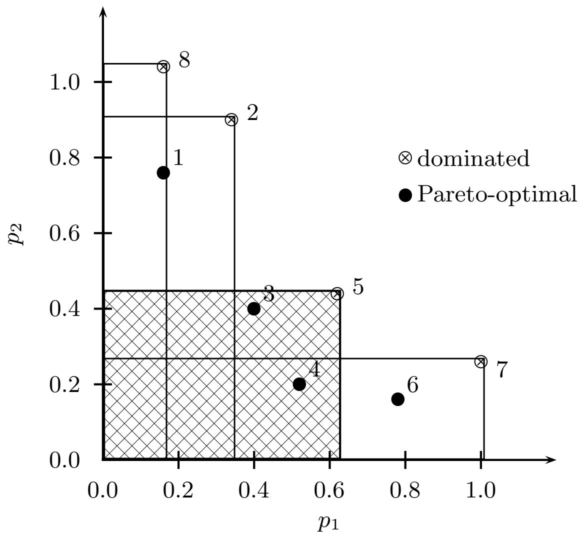

Figure 2: Example of solutions with two penalties. The two coordinates show the values of preferences p_1 (horizontal) and p_2 (vertical). Rectangles show where dominating solutions must fall. For example, solutions dominating solution 5 must fall into the hatched rectangle, so 5 is dominated by 3 and 4. Note also that solution 4 is optimal in the utilitarian model, while 3 is optimal in the egalitarian model.

We assume that the set of solutions displayed to the user, called the *displayed set*, is the union of two sets computed with different purposes. The first, called the *elicitation set*, is chosen to make the user aware of the variety of available solutions and stimulate expression of preferences. We will present an analysis that results in a criterion that shows what solutions to show to optimally stimulate expression of preferences.

The second, called the *solution set*, is chosen to allow the user to select a final solution. We present an analysis of different models for computing the optimal solutions and thus derive bounds on how many solutions have to be shown to guarantee that the optimal one is found.

3. STIMULATING EXPRESSION OF PREFERENCES

Behavioral decision theory ([11]) has studied the actual behavior of human decision makers and has repeatedly pointed out the adaptive and constructive nature of human decision making. Studies have shown in particular that user-involved preference construction is likely to be more effective than using default or implicit models if a user is to understand and accept the solution outcomes ([3]). Example critiquing can best support such an incremental construction process by providing examples that stimulate the user to express their hidden preferences.

We assume that users are stimulated for formulate a preference when either

- they see an example where this preference is violated, or

- they see a solution that satisfies that preference better than others and could be considered their most preferred choice.

The first case does not require showing any specific examples; if a currently shown example corresponds to all of the user's preferences he will choose it as a solution and terminate the process.

For the second case, the best solutions to show are those that have a high potential to become Pareto-optimal if an additional preference is stated. Let solution s_1 be dominated by a set S_d of other solutions. In order for s_1 to become Pareto-optimal, the user must add preferences that will eliminate all the dominance relations with other solutions in S_d. If we assume that the probability of this happening for a hidden preference is p_d for all $s_i \in S_d$, then the probability of s_1 becoming Pareto-optimal is proportional to:

$$\prod_{s_i \in S_d} p_d$$

If all we are interested in is an ordering of all dominated solutions according to this criterion, it is sufficient to compare the logarithms of this probability:

$$log(\prod_{s_i \in S_d} p_d) = \sum_{s_i \in S_d} log(p_d) = |S_d| log(p_d)$$

For ranking the solutions, we do not need to know $log(p_d)$, but it is sufficient to order them by the number of dominating solutions only. We call this a *counting* filter.

A more precise filter can be obtained by a more detailed analysis of the differences between solutions. Let A_u be the set of attributes on which no preference has been expressed yet, and assume that new preferences will only be expressed on this set. Then solution s_1 can only become Pareto-optimal if there is an attribute $a_i \in A_u$ such that all currently dominating solutions $s_d \in S_d$ have a different value for that attribute:

$$diff(a_i, s_1, S_d) = \begin{cases} 1 & \text{if } (\forall s_d \in S_d) a_i(s_1) \neq a_i(s_d) \\ 0 & \text{otherwise} \end{cases}$$

For an attribute a_i which is continuous and ordered, we can assume that preferences will select intervals of the attribute values. In this case, s_1 can become Pareto-optimal through a_i if and only if it falls outside of the interval of values for the dominating solutions S_d:

$$diff(a_i, s_1, S_d) = \begin{cases} 1 \text{ if} & (a_i(s_1) < min_{s_d \in S_d} a_i(s_d)) \vee \\ & (a_i(s_1) > max_{s_d \in S_d} a_i(s_d)) \\ 0 & \text{otherwise} \end{cases}$$

Letting α be the probability that the user will still place an additional preference on an attribute in A_u, and assuming that this is equally likely for all attributes, we have for the probability that s_1 could become Pareto-optimal:

$$\begin{aligned} p(s_1 \text{becomes P.-o.}) &= 1 - \prod_{a_i \in A_u} (1 - \alpha \cdot diff(a_i, s_1, S_d)) \\ &\approx \alpha \sum_{a_i \in A_u} diff(a_i, s_1, S_d)) \end{aligned}$$

By ordering the solutions according to this probability, we obtain what we call the *attribute filter*. As discussed below, this filter works very well for predicting what examples might become Pareto-optimal when just one extra preference is added, but performs more poorly when several extra preferences might still be posted.

To take into account the effect of a sequence of newly posted preferences, we can exploit the fact that for a solution to become Pareto-optimal, it must have lower penalty for the new preference than any solution that is dominating it. This is similar to the counting filter, except that we now estimate a different p_d for each dominating $s \in S_d$. This probability, which we call $p_d(s_1, s)$, is the probability that s will no longer dominate s_1 after adding a new preference.

For continuous attributes, we assume that penalty functions are monotonic from a certain threshold. Hence, solutions will behave differently with respect to a new preference if its threshold falls in the interval between them. The probability of this happening is proportional to the distance between them, i.e.:

$$p_d(s_1, s) \approx \sum_{a_i \in A_u} \beta(a_i) |a_i(s_1) - a_i(s)| P_{a_i}$$

where $\beta(a_i)$ should be chosen to normalize the range of each attribute, for example $\beta_i = 1/range(a_i)$, where $range(a_i)$ is the maximum range of values that attribute a_i takes. The a-priori probability that user will pose the new preference on a particular attribute, P_{a_i}, can be estimated through user studies or simply set equal to $1/|A_u|$.

For discrete attributes, we do not have an order, so we introduce a discrete distance:

$$|a_i(s_1) - a_i(s)| = \begin{cases} 0 & \text{if } a_i(s_1) = a_i(s) \\ 1 & \text{otherwise} \end{cases}$$

We can now express the probability that s_1 will become Pareto-optimal as:

$$\begin{aligned} p(s_1 \text{becomes P.-o.}) &= \prod_{s \in S_d} p_d(s_1, s) \\ &= \prod_{s \in S_d} \sum_{a_i \in A_u} P_{a_i} \beta(a_i) |a_i(s_1) - a_i(s)| \end{aligned}$$

and should then order solutions so that those with the highest probability come first.

Rather than computing products, it is often easier to compute a sum. This can be done by ordering the solutions by the negative logarithm of this probability, i.e. using the function:

$$\begin{aligned} F(s_1) &= -log(p(s_1 \text{becomes P.-o.})) \\ &= -log \prod_{s \in S_d} \sum_{a_i \in A_u} \beta(a_i) |a_i(s_1) - a_i(s)| P_{a_i} \\ &= -\sum_{s \in S_d} log \sum_{a_i \in A_u} P_{a_i} \beta(a_i) |a_i(s_1) - a_i(s)| \end{aligned}$$

This makes the computation similar to that of the counting filter, and we can see that the added complexity lies in the fact that we need to also determine the degree to which a solution is dominated by another. We call this the *probabilistic filter*.

To test the performance of the three methods, we have used a list of actual 44 apartment offers modelled by 11 attributes, of which 4 are continuous. On these attributes, we randomly generated preference models of 7 user preferences and a random order of specifying these preferences. We then

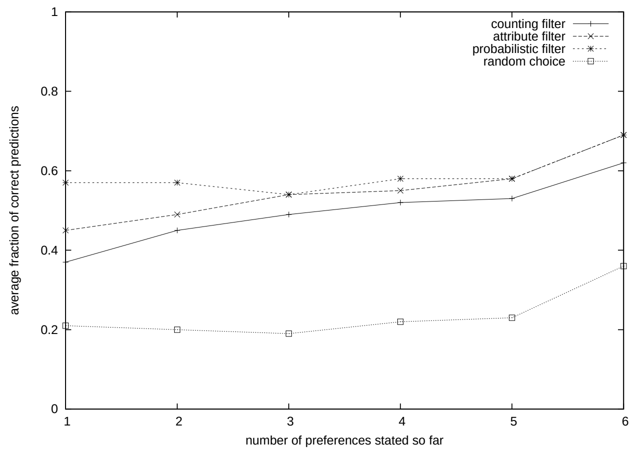

Figure 3: *Fraction of top-ranked 5 solutions that actually become Pareto-optimal as a function of the number of preferences already stated. We compare: counting the number of dominating solutions, counting the attributes that might lead to Pareto-optimality, the probabilistic analysis of the attribute differences, and randomly choosing a solution among those that are not yet Pareto-optimal.*

examined how well this filter would predict the solutions that would eventually become Pareto-optimal. Figure 3 shows what fraction of the 5 top-ranked solutions according to this criterion actually become Pareto-optimal when all 7 preferences have been stated as a function of how many of these preferences have already been stated. For comparison, we also show the probability of a randomly chosen solution becoming Pareto-optimal. We can see that the filter based on counting dominating solutions already provides good predictive accuracy, and that better accuracy can be obtained by analyzing the differences in the attributes. However, the best prediction is obtained in the probabilistic analysis of attribute differences, particularly in the early stages of the process. This method is also computationally the most complex to implement. In all cases, the accuracy increases significantly the more preferences have already been stated. This is explained by the fact that the more preferences have been stated, the more dominance relations exist to be used by the filter.

4. DECISION SUPPORT WITH STANDARDIZED PREFERENCE MODELS

The second major design issue we address in this paper is how to compensate for the inaccuracies caused by using a standardized and partial model of the user's preferences. In example critiquing, we compensate for this shortcoming by letting the user pick the most preferred solution from a larger *displayed set D* of k solutions. Such an approach has been taken in numerous practical systems, for example in [10, 12, 16, 2, 13, 17]. The process will be sound, i.e. allow the user to find the target solution, only if the displayed set actually contains the solution. We will now show that this heavily depends on the model used for selecting displayed solutions as well as on the number of preferences that have been stated.

For the example critiquing approach to be practical and successful, the following conditions must be satisfied:

1. it must be possible to limit the computed set D to a size of at most k displayed solutions so that it can be displayed in a consistent manner.
2. solutions that are Pareto-optimal within the set D must also be Pareto-optimal with respect to the set of all feasible solutions. This is important to keep the user from unknowingly picking a dominated solution as the final choice.
3. when the user has specified his preferences, the target (user's most preferred) solution must be included in the displayed solutions. We call this property *soundness* and it is required so that the user can actually choose the final solution.

Depending on the characteristics of the application, different methods for selecting displayed solutions may be appropriate. In our work, we have analyzed three different kinds of filter:

- *dominance filters* that display k solutions that are not dominated by any other one.
- *utilitarian filters* that keep the k solutions with the lowest sum of preference violations, and
- *egalitarian filters* that keep the k solutions with the smallest maximum preference violation.

In a longer paper ([5]), we have developed a theoretical model for the behavior of each filter, and validated the model using experiments with randomly generated data. Here, we summarize the models and the main results we obtained from them.

In order to allow a theoretical comparison, we assume that preferences p_i are independent, with real-numbered values in the interval [0..1], and that m solutions are distributed uniformly in the $|P^*|$-dimensional space of preference combinations. While in reality both assumptions are not likely to hold perfectly, comparing the theoretical results to measurements on real-world configuration problems shows quite a good match with reality.

4.1 Dominance filter

In the dominance filter, we filter out all dominated solutions and keep only those that are Pareto-optimal. A probabilistic analysis of this filter can be obtained as follows.

As shown in Figure 2, a solution s_j with preference values $p_1 = c_1, ..., p_d = c_d$ is dominated by any solution that falls within the subspace $p_1 \in [0..c_1], ..., p_d \in [0..c_d]$, which is a hypercube. Given a uniform distribution of solutions, the average probability that a solution s_j is dominated by another solution s_l is thus equal to the probability that s_l falls into that subspace. This probability is given by the proportion of the subspace with respect to the entire space. Since we assume all attributes to vary between 0 and 1, the size of the entire space is $1^d = 1$, so that the probability is just the size of the subspace, i.e.:

$$Pr(s_j \prec s_l) = \prod_{p_i \in P} p_i(s_j)$$

and the probability that a solution s_j is Pareto-optimal is the probability that in the $m-1$ other solutions, not a single

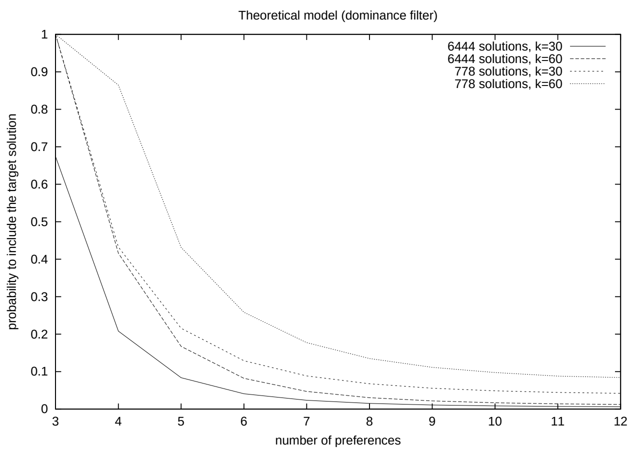

Figure 4: Probability that the target solution is actually in the displayed set for different numbers of preferences.

one dominates s_j:

$$Pr(s_j \text{ is P.-o.}) = \left(1 - \prod_{p_i \in P} p_i(s_j)\right)^{m-1}$$

The expected number of Pareto-optimal solutions is given by integrating this probability over the possible value combinations of the preferences, thus computing the expected size of the set $O_p = \{s_j | s_j \text{ is Pareto-optimal}\}$ of Pareto-optimal solutions:

$$E[|O_p|] = (m-1)\int_0^1 \cdots \int_0^1 \left(1 - \prod_{p_i \in P} p_i\right)^{m-1} dp_1 \cdots dp_d$$

where $d = |P|$ is the number of preferences in the set P.

This expression unfortunately does not have a closed-form solution, but when evaluated numerically it shows a rapid rise in the number of Pareto-optimal solutions as the number of criteria increases; this rise is confirmed by practical experiments. A consequence of this rapid rise in the number of Pareto-optimal solutions is that when there are few preferences, there may not be enough solutions to fill the displayed set, while with too many preferences, there will be too many solutions. Thus, in order to satisfy condition (1), the filter may have to perform random sampling.

Condition (2) is trivially satisfied as only solutions that are Pareto-optimal in the entire set of feasible solutions are shown.

For condition (3), we assume that the target solution s_t is Pareto-optimal. However, as already mentioned, it will often be necessary to make a random selection of solutions to actually display.

Figure 4 shows a plot of the probability of the target solution being included in the displayed set of k randomly selected Pareto-optimal solutions as a function of the number of preferences $|P| = 3, \ldots, 12$ for $m = 778, k = 30$ and $m = 6,444, k = 60$. We can see that the probability of including the target solution rapidly decreases as the overall set of Pareto-optimal solutions becomes too large and the target often is no longer selected, even for a relatively small number of preferences. Thus, the method does not satisfy condition (3) very well.

4.2 Utilitarian filter

This model is tailored to the case where the user's true preference model is utilitarian, i.e. by minimzing a quasilinear cost function $C^*(s) = \sum_{p_i^* \in P^*} p_i^*(s)$.

In the utilitarian filter, we approximate this by the predefined normalized penalty functions and choose as displayed solutions the k best solutions according to the unweighted sum of the penalties:

$$C(s) = \sum_{p_i \in P} p_i(s)$$

This set can be generated efficiently using branch-and-bound or other optimization algorithms (see [8]); in fact it can often be integrated with the generation of the feasible set itself.

The method obviously satisfies condition (1).

The following Theorem shows that it also satisfies condition (2):

THEOREM 1. *Given a set of m solutions* $\mathcal{S} = \{s_1, \ldots, s_m\}$ *and a set of d preference penalties* $\{p_1, \ldots, p_d\}$. *Let* $\mathcal{S}' = \{s_{i_1}, \ldots, s_{i_k}\} \subseteq \mathcal{S}$ *be the best k solutions according to the utilitarian filter:* $\forall s' \in \mathcal{S}', \forall s \notin \mathcal{S}' : C(s') \leq C(s)$.

If $s_x \in \mathcal{S}'$ *and* s_x *is not dominated by any other solution* $s_y \in \mathcal{S}'$, *then* s_x *is Pareto-optimal in* $\mathcal{S}$.

PROOF. *Assume that* s_x *is not Pareto-optimal in* $\mathcal{S}$. *Then, there is a solution* $s_z \notin \mathcal{S}'$ *which dominates solution* s_x, *and by definition:*

$$\begin{aligned} \forall p_i,\, p_i(s_z) &\leq p_i(s_x) \text{ and} \\ \exists p_j,\, p_j(s_z) &< p_j(s_x) \end{aligned}$$

As a consequence, we also have:

$$\sum_{i=1}^{d} p_i(s_z) < \sum_{i=1}^{d} p_i(s_x)$$

and therefore $C(s_z) < C(s_x)$. *But this contradicts the fact that* $s_x \in \mathcal{S}'$ *and* $s_z \notin \mathcal{S}'$. □

Thus, the method also satisfies condition (2).

To understand how far the method satisfies condition (3), we summarize the results of an analysis presented in detail in [5]. It is based on the observation that the inaccuracies in the penalty functions:

$$(1-\epsilon)p_i \leq p_i^* \leq (1+\epsilon)p_i$$

translate into a corresponding inaccuracy in the sum of these penalties:

$$C(s_t) \leq \frac{1+\epsilon}{1-\epsilon} C(s_1)$$

Through a probabilistic analysis that determines the expected sum of penalties for the k-th best solution, we can derive the expected ratio of the penalty sums of the k-th best to the best solution:

$$C_k / C_1 = k^{1/d}$$

which somewhat surprisingly is independent of the total number of solutions m and then derive the expected worst-case position t of the target solution in a solution list ranked by the penalty sums:

$$t = \frac{1+\epsilon}{1-\epsilon}^{\;d}$$

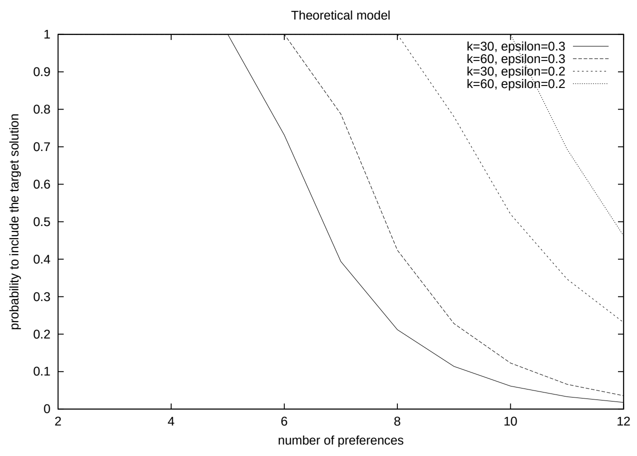

Figure 5: Probability that the target solution is within the displayed set.

This function can be used to calculate the number of solutions that have to be displayed for a given number of preferences and a given tolerance ϵ for user diversity, a crucial design parameter for example-critiquing interfaces.

For a particular design, it can also be used to show the probability that the target solution will be actually displayed. Figure 5 shows a plot of this probability for $|P| = 2, \ldots, 12$ and combinations of $\epsilon = 0.2, 0.3$ and $k = 30, 60$. We can see that with each design, there is a sharp phase transition where the performance of the decision support system degrades sharply. This shows the importance of careful analysis and design in order to not mislead the user.

4.3 Egalitarian filter

The egalitarian filter is designed for the case where the user's true preference model is the egalitarian model, i.e. he minimizes the maximum penalty $F^*(s) = max_{p_i^* \in P^*} p_i^*(s)$. The egalitarian filter approximates this by applying the same criterion on the standardized penalty functions:

$$F(s) = max_{p_i \in P} p_i(s)$$

where ties among several solutions with the same maximal value are broken by applying the same criterion to the remaining preferences. The method corresponds to the decision criteria used in fuzzy logic, and has the advantage that it can be efficiently implemented using fuzzy constraint satisfaction techniques.

Similarly to the utilitarian filter, we assume that the penalty functions are normalized for a standard user, but that for a particular user they can deviate by no more than a factor of ϵ in either direction.

This method obviously satisfies condition (1), as it generates exactly k solutions.

It also satisfies condition (2), since we have the following Theorem:

THEOREM 2. *Given a set of m solutions $\mathcal{S} = \{s_1, \ldots, s_m\}$ and a set of d preference penalties $\{p_1, \ldots, p_d\}$. Let $\mathcal{S}' = \{s_{i_1}, \ldots, s_{i_k}\} \subseteq \mathcal{S}$ be the best k solutions according to the egalitarian filter.*

If $s_x \in \mathcal{S}'$ and s_x is not dominated by any other solution $s_y \in \mathcal{S}'$, then s_x is Pareto-optimal in $\mathcal{S}$.

PROOF. *Assume that s_x is not Pareto-optimal in $\mathcal{S}$. Then, there is a solution $s_z \notin \mathcal{S}'$ which dominates solution s_x, and by definition:*

$$\begin{aligned} \forall p_i,\, p_i(s_z) &\leq p_i(s_x) \;\; \textit{and} \\ \exists p_j,\, p_j(s_z) &< p_j(s_x) \end{aligned}$$

Let p_m be a preference with the highest penalty in solution s_x:

$$\forall p_i,\, p_i(s_x) \leq p_m(s_x)$$

Then since s_z dominates s_x, $p_m(s_z) \leq p_m(s_x)$ and $\forall p_i, p_i(s_z) \leq p_m(s_z)$ and thus $F(s_z) \leq F(s_x)$.

When $F(s_z) < F(s_x)$, this contradicts the fact that $s_x \in \mathcal{S}'$ and $s_z \notin \mathcal{S}'$.

When $F(s_z) = F(s_x)$, we have a tie and the same argument applies to the set of preferences with p_m removed. As s_z dominates s_x, there must eventually be a preference p_e such that $p_e(s_z) < p_e(s_x)$, leading to the contradiction as above. $\square$

As for condition (3), a similar analysis as for the utilitarian filter gives us the following result:

$$t \geq \left(\frac{1+\epsilon}{1-\epsilon}\right)^{(d-1)}$$

This gives us in fact a similar behavior as for the utilitarian filter, but the egalitarian filter allows one more preference with the same number of displayed solutions.

4.4 Robustness against violated assumptions

In our analysis of the utilitarian and egalitarian filters, we have assumed that the user's preference model actually follows this model. However, it may be that this assumption is false, and that the only thing that can be assumed of a particular user is that he will not prefer a dominated solution.

The performance of the filter in this case can be characterized by the number of solutions that a given Pareto-optimal solution s_t will be within the k best solutions found by each of the filters.

Following an analysis presented in more detail in [5], we obtain the results shown in Figure 6, which are for a problem with $m = 6'444$ solutions when $k = 30$ and $k = 60$ solutions are shown for different numbers of preferences. We can see that for both the utilitarian and the egalitarian filter, the probability decreases quite dramatically; however, the utilitarian filter is in general slightly better than the egalitarian filter.

When comparing with Figure 4, we see that in fact the performance of utilitarian and egalitarian filters is almost as good as that of a dominance filter. On the other hand, implementing the dominance filter would use algorithms for computing all Pareto-optimal solutions, and these are extremely inefficient. Thus, even when the user's true preference model is unknown, it is likely to be better to use a utilitarian filter instead.

In a similar analysis that we omit for reasons of space, we can show that the utilitarian filter behaves poorly when the true preference model is an egalitarian one, and the egalitarian filter behaves even worse when the true preference model is utilitarian.

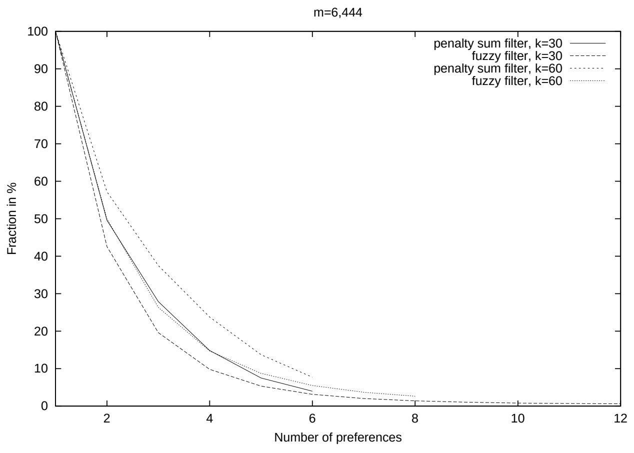

Figure 6: Fraction of Pareto-optimal solutions shown within the k best ones according to utilitarian and egalitarian filters.

5. CONCLUSIONS

Example critiquing has been shown to be a useful paradigm for user interaction with intelligent decision aids. So far, work has focussed on systems that demonstrated the usefulness of the concept in particular domains. In this paper, we have started to put example critiquing on firmer ground by presenting mathematical analyses of how such systems should be designed.

The ingenuity of example critiquing is that it elegantly combines an efficient method for eliciting the user's preferences with a way to compensate for inaccuracies in this preference model by showing a multitude of best solutions. We have analyzed the two major design issues that arise in this context: what examples to show to stimulate expression of hidden preferences, and what and how many solutions to show to compensate for the inaccurate preference model.

Mathematical decision theory has so far largely overlooked the challenges posed by the complex human-computer interaction that would be required to apply it correctly. In this paper, we have shown how decision theory can be adapted to provide principles of example critiquing systems.

Preliminary user studies and observations on a commercial product validate the principles we obtained through the mathematical model. For example, users always felt that showing only 5 solutions as in the ATA travel planning tool ([10]) was insufficient. Our analysis shows that indeed, 5 solutions are unlikely to be enough on typical instances of the travel planning task, and 30 is a more appropriate number. We hope that our analysis will enable more rapid design of successful example-critiquing systems in other areas as well.

6. REFERENCES

[1] C. Boutilier, R. Brafman, C. Geib, and D. Poole: "A Constraint-Based Approach to Preference Elicitation and Decision Making," AAAI Spring Symposium on Qualitative Decision Theory, Stanford, 1997

[2] R. Burke, K. Hammond, and B. Young: "The findme approach to assisted browsing," IEEE Expert **12**(4), pp. 32-40, 1997

[3] G. Carenini and D. Poole: "Constructed Preferences and Value-focused Thinking: Implications for AI research on Preference Elicitation," AAAI-02 Workshop on Preferences in AI and CP: symbolic approaches - Edmonton, Canada, 2002.

[4] J. Doyle and R. Thomason: "Background to qualitative decision theory," AI Magazine **20**(2), Summer 1999

[5] B. Faltings, M. Torrens and P. Pu: "Solution generation with qualitative models of preferences," to appear in *Computational Intelligence*, special issue on qualitative decision theory, 2004, available at `http:\\liawww.epfl.ch\~faltings\compint-2004.pdf`

[6] V. Ha and P. Haddawy: "Problem-focused incremental elicitation of multiattribute utility models," 13th Conference on Uncertainty in Artificial Intelligence, pp. 215–222, August 1997

[7] V. Ha and P. Haddawy: "Toward Case-Based Preference Elicitation: Similarity Measures on Preference Structures," 14th Conference on Uncertainty in Artificial Intelligence. 1998

[8] R.Keeney and H. Raiffa: "Decision with multiple objectives: Preferences and value tradeoffs," Cambridge University Press, 1976

[9] R. Keeney: "Value-Focused Thinking: A Path to Creative Decision Making," Harvard University Press, 1992

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

[11] J.W. Payne, J.R. Bettman and E.J. Johnson: "The Adaptive Decision Maker," Cambridge University Press, 1993

[12] P. Pu and B. Faltings: "Enriching Buyers' experiences: the SmartClient Approach," ACM SIGCHI 2000 Conference on Human Factors in Computing Systems, pp. 289-296, 2000

[13] S. Shearin and H. Lieberman: "Intelligent Profiling by Example," International Conference on Intelligent User Interfaces (IUI 2001), pp. 145-152, 2001

[14] M. Stolze: "Soft Navigation in electronic product catalogs," International Journal on Digital Libraries 3(1), pp. 60-66, 2000

[15] M. Torrens, R. Weigel and B. Faltings: "Java Constraint Library: bringing constraint technology on the Internet using the Java language," Workshop on Constraints and Agents (AAAI97), AAAI Technical Report WS-97-05, July 1997

[16] M. Torrens, B. Faltings and P. Pu: "SmartClients: Constraint Satisfaction as a Paradigm for Scaleable Intelligent Information Systems," CONSTRAINTS **7**, p. 49-69, 2002

[17] M. Torrens, P. Hertzog, L. Samson and B. Faltings: "*reality*: a Scalable Intelligent Travel Planer, Decision Support for the Business Traveler," 18th ACM Symposium on Applied Computing, Melbourne, 2003