A Survey of Preference Elicitation

Technical Report TR-2005-41
Brent M. Dennis, Christopher G. Healey

Knowledge Discovery Lab
Department of Computer Science, North Carolina State University
Raleigh, NC 27695-8207

Email: bmdennis@unity.ncsu.edu

Abstract

As more information becomes available to users, there is an increasing number of complex tasks that motivate the need for computer-assisted interaction, such as the exploration of a large multidimensional information space. In order to assist users with achieving their goals, computers need to accurately model the preferences of their users. Preference elicitation techniques attempt to collect important information about the user's preferences and interests. This survey describes various preference elicitation techniques and systems. Furthermore, the survey explores the potential for integrating preference elicitation techniques into an existing assisted navigation system designed to aid users with data exploration. The navigation system assists users by identifying and locating interesting elements within the visualization of a multidimensional dataset. The survey will also discuss how the incorporation of preference elicitation techniques could potentially benefit the navigation assistant.

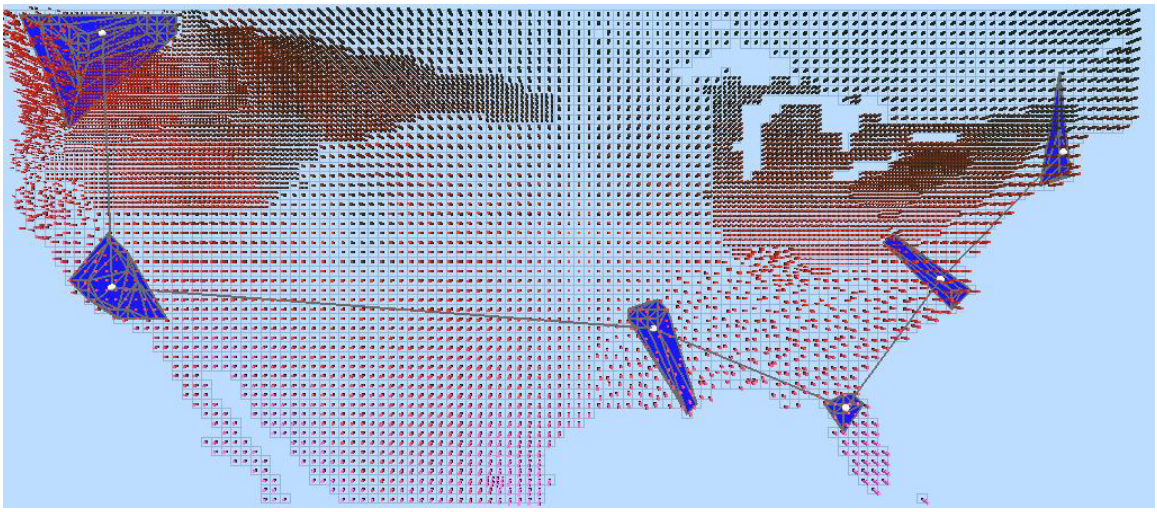

Figure 1: A visualization of a weather dataset composed of perceptual texture elements (or *pexels*). Dense areas represent high cloud cover, irregular areas represent high precipitation, and red and pink colors correspond to high temperature values. Elements of interests have been identified and clustered (shown in blue). The underlying navigational graph framework is also shown.

1 Introduction

As computers play a larger role in the lives of their users, it is becoming increasingly important for computers to understand and represent users' preferences. Today there is an abundance of complex tasks that motivate the need for mixed-initiative and computer-aided interaction. Examples of such tasks include managing e-commerce interactions such as auctions and elaborate planning tasks such as scheduling a vacation.

With the increasing amount of information that can be collected, the development of more systems to assist with data exploration and knowledge discovery is a high priority. The area of visualization provides techniques to create meaningful graphical displays of information from which users may build a deeper knowledge and understanding about a dataset. However, for many modern multidimensional datasets, recent visualization systems are unable to create a comprehensible representation that can be displayed as a single, unified image on a computer screen.

Navigation concepts were combined with scientific and information visualization techniques to build a navigation assistant to aid users with finding interesting elements within a multidimensional dataset. Its goal was to help users seek out and explore data that was located "offscreen," *i.e.* beyond the user's current field of view [DH02]. Interesting elements were explicitly identified *a priori* by the user with a set of interest rules built using a simple boolean and mathematical grammar. The navigation assistant then tagged these elements of interest and built an underlying framework in order to create automated camera paths between them. Figure 1 shows the visualization of a weather dataset with elements of interest identified and incorporated into the navigation assistant's graph framework. Figure 1 also shows how the elements of interest were then clustered together to form areas of interest. Using the graph framework,

the navigation assistant can also construct automated tours between or within areas of interest.

This scheme should be improved on for several reasons. First, it is potentially difficult for users to accurately describe their interests using the above language. Second, the users' interests may change over time or during the exploration process thus requiring a halt of the exploration process in order to revise the set of interest rules. Similarly, users may discover new interests during the exploration process which should be incorporated into the existing rule base without interrupting exploration. Finally, analyzing existing user interests may enable the assistant to build additional rules describing new and unexpected elements or trends the user may wish to explore. By observing how the user interacts with the visualization, we hope to develop techniques for the *implicit* identification of interesting elements. We hope to better integrate the exploration process with the task of maintaining a model of the user's interests. Determining which elements of a dataset are interesting to a user is a task that seems closely related to determining the preferences of the user.

The acquisition of user preferences is the goal of *preference elicitation*. Decision support systems rely on preference elicitation to provide them with accurate and useful information for the construction of accurate and effective user models. As opposed to other existing techniques such as collaborative filtering, preference elicitation makes no assumptions about the existence of any initial user information. If no information is available at the start of interaction, preference elicitation must attempt to quickly collect as much preference data as possible so that users can begin working toward their goals. Furthermore, they must also be able to resolve potential conflicting preferences, discover hidden preferences, and make reasonable decisions about tradeoffs with competing user goals.

2 Decision Theory

The goal of preference elicitation is to facilitate the construction of an accurate user model that can be used by a decision support system to assist a user with the completion of a task. Preference elicitation is usually designed to provide the necessary data for a particular decision support framework. The theoretical foundation that forms the basis of these models is found in decision and utility theory [KR76, Fre86]. Decision and multi-attribute utility theory focus on the evaluation of choices and outcomes for a decision problem or scenario.

Outcomes are defined by the assignment of values to a set of attribute variables, $X = \{X_1, ..., X_n\}$. Attribute variables are either discrete, $x_i \in \{x_{i,1}, ..., x_{i,m}\}$, or continuous, $x_i \in [a, b]$. The set of outcomes O considered for a decision problem is contained by the *outcome space* Ω which is defined by the Cartesian product of X. Thus, $O \subseteq \Omega$ where $\Omega = \{X_1 \times X_2 \times ... \times X_n\}$. It is common for O to be very large. Even an outcome space composed of discrete attribute variables can potentially be combinatorially large. It is often the case that Ω contains outcomes that are infeasible with respect to the current decision problem. Consider an allocation decision problem composed of two attribute variables, $X = \{X_\alpha, X_\beta\}$, each variable corresponding to one of two separate auction agents, α and β. The value of one of these variables is a possible assignment of goods from a set of goods, $\{A, B\}$, to a particular agent, $x_\alpha \in \{\emptyset, A, B, AB\}$ where x_α is a particular assignment for the attribute variable X_α.

An outcome in Ω is of the form $o = \{x_\alpha, x_\beta\}$ and a potential value could be $o = \{\emptyset, AB\}$, corresponding to agent α receiving nothing and agent β receiving everything. Obviously, any allocation where both agents acquire the same good is infeasible (*e.g.* $o = \{A, AB\}$) and should not be considered by a decision maker. However, in practice, Ω remains very large even with the removal of infeasible outcomes.

In order to make decisions based on a set of outcomes O, a decision maker often simply needs an ordering of the outcomes determined by the user's preferences. This is called a *preference relation* and is denoted by $\succeq$. Given $o_i, o_j \in O$, if $o_i \succeq o_j$, o_i *dominates* or is *preferred* to o_j. Typically, the preference relation is induced by a *value function*, $v(o) : O \rightarrow \mathbb{R}$. For a given value function v and its induced preference relation, $\succeq$, the following is true,

$$\forall o_a, o_b \in O,\ o_a \succeq o_b \iff v(o_a) \geq v(o_b).$$

Value functions can operate on the set of outcomes or the attribute variables themselves. The value for a given attribute variable dominates another value if the following holds true,

$$\forall a, b \in X_i,\ a \succeq b \iff v(a) \geq v(b).$$

Value functions reflect how much a user values acquiring a particular outcome. However, in many decision scenarios there may exist a degree of uncertainty. Performing a given action could result in achieving outcome o_1 with probability p_1 or achieving outcome o_2 with probability p_2. In cases of uncertainty, the value function alone is not enough to make appropriate decisions. Since a particular action no longer guarantees a particular outcome, a more complex function is needed to evaluate the "utility" of a decision. Utility theory provides techniques that will incorporate the user's attitudes about risk [KR76].

A main contribution of utility theory is a theorem which proves the existence of a *utility function*, $u(x) : O \rightarrow \mathbb{R}$, that will induce a unique preference relation, $\succeq$, over O. Unfortunately, a preference relation can not induce a utility function over the set of outcomes because the utility function must also account for the user's attitudes about risk. The utility function is often confused with the value function and the literature will often interchange their names. In decision theory, the utility function represents the user's attitudes about risk as well as the user's value of the outcomes. Therefore, it induces a preference ordering on the probability distributions (or *lotteries*) over the outcome space. If Pr_i and Pr_j correspond to the two probability distributions over the outcomes by the invoking of two actions a_i and a_j then

$$a_i \succeq a_j \iff \sum_{o \in O} Pr_i(o)u(o) \geq \sum_{o \in O} Pr_j(o)u(o)$$

for the utility function u. When assigning values for an outcome o, the utility function u must consider the uncertainty of attaining o and the user's attitudes toward risk to correctly preserve the user's preference relation for actions. The above relation implies that users prefer the action resulting with the *maximum expected utility* (MEU). The utility function is important to elicitor systems because they assume that the user is *rational*, *i.e.* the user prefers the outcome with the maximum expected utility over all other outcomes. For a given situation, there could exist a single utility function for a group of users or each user might require a unique utility function.

Since at the start of interaction the utility function for the user is often unknown, constructing the utility function is the initial goal of many decision support systems. However, this is a difficult task since a complete description of a user's preferences is often not available or practical to attain. Acquiring this description for building an accurate utility function and preference relation for a user is the main goal of preference elicitation.

3 Challenges of Elicitation

Despite the difficulties of building an accurate and effective user model, eliciting preference information from a user is itself a non-trivial task. Many considerations should be made by a system, or *elicitor*, during the elicitation of preference information. Otherwise, systems may erroneously model the user, collect false information, or unintentionally influence the user's preferences [BF95].

Having a user completely describe his preference order upon initial interaction is often impractical. First, the number of potential pairs of outcomes for comparison by the user is a function of the size of the outcome space. In the best case, an outcome space with discrete variables, the outcome space grows combinatorially with respect to the number of attribute variables and values. Outcome spaces composed of even a small number of variables can generate far too many possible pairs for users to compare, requiring excessive amounts of time and effort on a user's behalf. This consideration assumes that infeasible outcomes have previously been removed from the set of outcomes for the user's consideration. The presence of infeasible outcomes only increases the amount of work required of the user. Research has shown that acquiring the right partial preference information can be nearly, if not just as, successful in finding optimal outcomes [HS02, HH99].

Furthermore, preferences are not always fixed. It is very common that different users have different preferences. There are some scenarios where there exist common preferences that can be used as a reference or starting point for elicitation. Elictors must be prepared to deal with each user on an individual basis. Another important factor is users might change their preferences during interaction with the system. As the user performs a task, old preferences may give way to new ones. Elicitors should be able to determine when this occurs and make the appropriate adjustments.

Pu *et al* describe another issue that should be addressed by preference elicitation techniques [PFT03]. Given the elicitation scheme, a particular choice of attributes may force users to state their preferences via *means* objectives instead of *fundamental* objectives. For example, consider a traveler who is looking to take a long weekend trip to New York. He wishes to restrict the amount of money he spends on airline tickets to only 300 dollars (his fundamental objective). If an elicitation scheme asks him how much he is willing to spend on his departure ticket, the traveler is forced to form a means objective. If he estimates to spend equal amounts of money on both departure and return flights, specifying 150 dollars would exclude a potentially optimal outcome of a 200 dollar flight into New York on Thursday afternoon paired with a 75 dollar red-eye return flight on Sunday night.

Furthermore, it has been shown that the method of elicitation can influence how users

respond to elicitation [BF95]. Users may unintentionally misinform the system about their preferences. Elicitors should elicit the user as infrequently as possible with queries that will provide the highest yield of preference information while simultaneously reducing any error about the user's preferences.

Incremental preference elicitation attempts to satisfy the above criteria by intelligently deciding how to next elicit the user based on information already collected. Incremental elicitation techniques may or may not have any initial preference information about the user. This is often the case with many scenarios. Incremental elicitation will also allow users to revise their preferences as they interact with the decision support system. Such capability is not possible with a system that requires full elicitation from the beginning or during a decision task.

4 Types of Queries

In order to improve the navigation assistant in [DH02], integrating new techniques for acquiring user preferences is a primary goal. In order to emphasize the exploration process, we plan to reduce the amount explicit responses to preference queries by the user. Instead, we hope to acquire preference information from the user by the observation of how the user interacts with the visualization. Is it possible to interpret the user's behavior in ways that will provide important preference information? For example, if the user spends significant time examining a particular element, does that imply the user is very interested in that element? Since querying the user is the primary mechanism for acquiring preference information for many elicitation systems, a review of many common types of queries is given below.

The first step to eliciting preference information is to determine the form of the questions submitted to the user. Elicitation queries should provide additional preference information that can allow an elicitor system to tighten its understanding of a user's preferences. Furthermore, elicitation queries should be simple to promote an accurate collection of information.

The most simplistic way to discover a user's preference relation is to task the user with pairwise comparisons of the possible outcomes. Such *order* queries may have the form of "Do you prefer o_a or o_b?" Many models assume that preferences are *transitive*: if $o_a \succeq o_b$ and $o_b \succeq o_c$ then $o_a \succeq o_c$. If transitivity holds across the user's preferences then order queries can reveal a great deal about preference. Otherwise, order queries are not an efficient elicitation query since they will only reveal local relationships among different outcomes. Another disadvantage to order queries is that they only provide an ordering of the outcomes. There is no qualitative metric to describe how much a user prefers one outcome to another. Order queries can only provide information about the user's preference relation and not the user's value or utility function. However, it is often easier for users to specify which outcome is preferred over another without having to compute an absolute and accurate numeric value for a particular outcome.

Rank queries are a more specific type of order query. They request the user to explicitly assign a rank to an outcome among all other outcomes. Rank queries may have the form "What is the rank of *a* or what outcome has rank *r*?" These type of queries are not always feasible since the set of outcomes may be too large for the user to produce accurate rank values. Used

with order queries, rank queries provide a helpful mechanism for approximating where a given outcome resides in the preference relation.

Value queries help provide a quantitative measure of preference between outcomes. They establish how much a user prefers one outcome to another. They might be of the form "What is the value for outcome *a*?" The ability of a user to truthfully answer a value query is often dependent on the size of the set of outcomes. It is difficult for a user to compute accurate values for individual outcomes among a large number of outcomes.

A variation of the value query is the *standard gamble* query. The standard gamble query measures how strong a user's preference is for an outcome by evaluating the risks the user is willing to make in order to attain this particular outcome [NM47]. The standard gamble query is common among decision theory research because it provides information that allows for very accurate modifications to a user's preference model. Consider three outcomes o_a, o_b, and o_c. The user's preference ordering has $o_a \succ o_b \succ o_c$. A standard gamble query asks if the user prefers to definitely acquire o_b or accept a gamble (or *lottery*) where the user acquires o_a with probability π or o_c with probability $(1 - \pi)$. Utility theory guarantees a value for π where the user is indifferent between acquiring o_b and attempting the gamble. The value of π for which the user is indifferent between the gamble and the outcome o_b determines the utility value of o_b: $u(o_b) = \pi u(o_a) + (1 - \pi)u(o_c)$. Typically, the best and worst case outcomes are used for o_a and o_b. If $u(o_{best}) = 1$ and $u(o_{worst}) = 0$ the aforementioned formula easily shows that π is the value of the utility function for o_b.

Potential outcomes or solutions are another form of queries used to determine the preferences of the user. This type of elicitation is often search-driven. Users move through a solution space evaluating possible candidates chosen by the system. How the user responds to potential candidates determines the selection of other potential optimal outcomes. Search is driven by having the user point out specific properties that are preferred or disliked from the given examples. Providing example solutions is already a part of the navigation assistant, although in a non-incremental fashion. Since the assistant functions in a visualization environment, the use of example queries integrates easily into the navigation assistant making it an ideal tool for elicitation.

Most of these query formats require explicit responses from the user. However, the majority of these queries may be potentially translated by the navigation assistant from direct questions to interpreted behavior. For the value query, the length of time spent examining a data element might be an indication of the user's value of that particular element. For order queries, the sequence a user decides to view potentially interesting elements which are in close proximity of one another might be an indication of which ones are more preferred to the others. Approximate rank information could be collected by considering how a user previously ignored a particular element only to return to it at a later time. Standard gamble queries are perhaps the hardest to translate into an observable behavior which is unfortunate given their efficiency for providing preference information.

5 Preference Structure

Given that the size of outcome spaces with only a few attributes can be potentially large, decision support systems must take advantage of any structure inherent to the user's preferences in order to facilitate an effective interaction between both the system and the user. Decision theory describes various forms of structure that can be found in preferences. Research has identified a variety of *independences* that potentially allows decision makers to consider the components of a given decision problem piecemeal [KR76, BG95]. In other words, identifying independence allows for the reduction of the number of outcomes for consideration by reducing the number of independent combinations of attributes. Independence also allows for the construction of less complicated and more structured utility functions. Determining the structure of the user's preference is potentially useful for the navigation assistant because it could provide insight about how certain attributes may affect user behavior towards a particular data element.

The most basic form of independence is *preferential independence*. Preferential independence considers only the preference ordering over the individual outcomes for a decision problem. A set of attributes $Y \subset X$ is *preferentially independent* of $X - Y$ when the preference order over outcomes with varying values of the attributes in Y does not change when the attributes of $X - Y$ are fixed to any value, *i.e.* the preference order of outcomes with attribute values in Y does not depend on the values of attributes in $X - Y$. Let α be an assignment of values to the attributes in Y and γ be an assignment of values to attributes in $X - Y$. If Y and $X - Y$ are preferentially independent then every $x \in X$ has the form $x = (\alpha, \gamma)$. Formally, preferential independence is defined as

$$\forall \gamma, \gamma' \in (X - Y) : (\alpha, \gamma) \succeq (\beta, \gamma) \iff (\alpha, \gamma') \succeq (\beta, \gamma')$$

with $\alpha, \beta \in Y$.

Utility independence is another form of independence concerned with the utility function as well as the user's preferences among the outcomes. Not only must the induced preference relation remain the same, but the user's attitude about risk or the relative strength of preference between outcomes must also remain the same. To understand utility independence, the notion of *conditional preference* must be described. For a conditional preference relation $\succeq$ with $Y \subset X$, let α and α' be two assignments of values to the attributes in Y. If $Z \subseteq X - Y$, let γ be an assignment of values to the variables in Z. α is *conditionally preferred to* α' *with respect to* γ if $\alpha \succeq \alpha' \iff (\alpha, \gamma) \succeq (\alpha', \gamma)$. Conditional preference can also be applied to probability distributions over outcomes. Using the same assignments for Y and Z described above, consider a probability distribution *Pr* over X. There exists a unique distribution Pr^{γ} such that the marginal probability of Pr^{γ} (the probability that the attributes of Z take the values of γ regardless of the values of the attributes in Y) over Y is *Pr* and Pr^{γ} gives the probability of 1 to the values of γ. Given a utility function with its associated preference order $\succeq$, the conditional preference over X given γ, $\succeq_{\gamma}$, is defined as

$$Pr_a \succeq_{\gamma} Pr_b \iff Pr_a^{\gamma} \succeq Pr_b^{\gamma}$$

for two probability distributions Pr_a and Pr_b over X.

With the definition of conditional preference established for probability distributions over the set of outcomes, a concrete notion of utility independence can be described. If $X = (Y, Z)$, Y is said to be utility independent of Z if the conditional preferences for distributions over Y given an assignment of values γ to the attributes in Z do not depend on the particular values of γ. Moreover, it is known that Y is utility independent of $X - Y$ if and only if the induced preference structure's utility function has the form,

$$u(X) = f(X - Y) + g(X - Y)h(Y)$$

for a positive function g [BG95]. Having a utility function of this form can be an improvement since the number of independent numbers to record is far fewer than the number required of preference structures with dependent structure.

A stronger independence can identified in a preference structure if the following condition is met. Let X be partitioned into $X_1, \ldots, X_k$ and let p_1 and p_2 be any two probability distributions that share the same marginal probabilities for X_i for all i. If p_1 and p_2 are indifferent over the preference structure ($p_1 \succeq p_2$ and $p_2 \succeq p_1$) then $X_1, \ldots, X_k$ are said to be *additive independent*. A set of variables for an outcome space which is additive independent has a utility function of the form

$$u(X) = \sum_{i=1}^{k} f_i(X_i)$$

where f_i could be considered the utility function for the attribute X_i or the *subutility functions*.

How independence among attributes contributes to preference elicitation is important to create more efficient elicitation techniques and interfaces. Given the large number of outcomes a decision problem can potentially create, independence among attributes allows elicitors to refine the set of potential queries needed to build an accurate representation of the user's preferences. For preference structures with additive independence, Keeney and Raiffa provide a procedure for reducing the number of queries by creating scales for each of the components of the utility function and querying the user about the behavior of each subutility function [KR76]. Variations of the standard gamble and value queries are then used to construct the appropriate scaling constants, k_i.

Chajewska and Koller work to construct more generalized factorizations of a utility function. They treat utility functions as random variables and create a statistical model of an expected utility function. Their model assumes that the population of users can be segmented into subpopulations. Members of the subpopulations typically have similar utility functions allowing the creation of a probability density function over a subset of the variables of X. If C is a set of clusters of variables from X, $C = \{C_1, \ldots, C_m\}$ where C_i and C_j for every i and j are not necessarily disjoint, a utility function is said to be *factored according to C* if there exists a function $u_i : Dom(C_i) \rightarrow \mathbb{R}$ such that the utility value of $o \in O$ is $u(o) = \sum_i u_i(c_i)$ where c_i is an assignment of values to the variables in C_i. These functions are the *subutility functions*. Note, that this "factorization" allows for multiple clusters to share variables in X, a condition not permitted with additive independence. The statistical model creates a subutility

function for each subpopulation and then uses this function for guiding the elicitation process. Unfortunately, this work depends on the existence of a pre-existing database of user utility functions.

Perhaps the most important contribution of independence from an intuitive aspect is that it allows the construction of simple and manageable utility functions. For additive independent attributes, the user's utility for any given outcome can be broken down to the sum of values for individual attributes. Instead of a utility function with n parameters, the system now has n utility functions with one parameter. Revision of the utility function takes place at the most atomic level: the attributes. It can be accomplished by manipulating scaling constants and the value of the utility function for a given attribute value. Changing the utility function with respect to one attribute has no impact on the other attributes' utility functions, allowing systems to revise one attribute at a time. Not only does this simplify interface design, but it is also easier for users to differentiate their values between two outcomes.

6 Preference Representation

The elicitation process often depends on how a user's preferences are represented in the system. The representation structure can determine what information is needed by the elicitor and the order that information should be collected. The representation of a user's preferences is usually associated with representing the user's utility function. Since most decision scenarios deal with multi-attribute decision problems, all of these representations are capable of expressing the preferences of a user over multidimensional datasets used by the navigation assistant. These representations tend to be either vector-based or graphically (or network) based.

Vectors are fundamentally simple representations of preferences. The content of vector representations depends on the particular elicitor. Some systems create a vector of length $|O|$ in which to store the values or utilities of every potential outcome. Evaluating the utility of a particular outcome involves referencing the correct index of the vector. In order to adjust the value of the utility function, the vector's values are updated accordingly. These types of implementations model the entire domain and range of the utility function [Bou02, CKP00, CGNS98, HH98]. This type of vector provides a determinate representation of the utility function, offering little information about how a utility value was computed for a given outcome. As a consequence of this, such vectors do not intuitively represent any structure that might exist within the preference relation. However, this is not to say that vector representations can not be used to discover preference structure as this was done in [CK00].

Sometimes vectors store components which are then used to compute the utility value of an outcome. The Automated Travel Assistant maintains a vector of constraints and a vector of weights. Each constraint is a function $C_i(x) : dom(X_i) \rightarrow [0, 1]$, where $C_i(x) = 0$ means the constraint is fully satisfied and $C_i(x) = 1$ means the constraint is fully unsatisfied. ATA makes the assumption that the preference structure's attributes are additive independent. The authors use these definitions to construct an *error* function which provides an antipodal utility function measuring how un-useful a given outcome is for a user,

$$error((x_1, \ldots, x_n)) = \sum_{i=1}^{n} w_i C_i(v_i)$$

where w_i is a weight ranging from [0, 1] for C_i. As the user provides additional information about the values of C_i and w_i, the vector is updated and the error function begins to approach zero, *i.e.* the constraints more accurately represent the user's preferences. ATA's representation is very similar to the navigation assistant's interest rule set. However, ATA allows for revisions and additions to the existing set of constraints during interaction. It is very possible to incorporate several of ATA's representation concepts into the rule scheme of the navigation assistant.

ATA models user preferences as a Constraint Solving Problem (CSP). CSP preference representations have been used in a variety of systems [BBGP97, BHY97, LHL97, SL01, TF99]. Once an appropriate set of constraints, C, has been acquired, constraint solving routines can determine potentially optimal solutions. If C is not tight enough to narrow the set of candidates, then additional constraints must be elicited from the user. In many traditional CSP solvers, constraints are *hard*, *i.e.* either fully satisfied or not. If no outcome can satisfy every constraint in C, CSP solvers resort to relaxing constraints until determining an optimal solution. Hard constraints alone are not adequate for describing preferences since it is possible for users to have preferences that conflict with one another. Ranking a set of hard constraints helps to assist with determining the user's importance of particular preferences and which constraints should be relaxed first. However, there could still be cases where a user's utility for a few highly-ranked constraints could be superseded by the utility of a larger number of lower-ranked constraints. This situation is hard to capture by simple ranking techniques. The use of *soft* constraints offers greater flexibility because soft constraints reflect how well an outcome satisfies a given constraint. The degree an outcome satisfies C is typically a sum of the products of an importance weight with a soft constraint value, very similar to the error function of ATA. When paired with importance weights, certain soft constraints can be emphasized over others.

Various types of network and graph structures are also used to represent user preferences [BG95, Bou01, CS01b, CS01a, HHR$^+$03]. An attractive feature about networks is that their topology lends itself to aid with the realization of structure in preferences and visually understand how outcomes come to dominate other outcomes. Graphical models can provide a more intuitive representation than do vectors.

Graphical models can support the representation of preference structures with *intransitive* structure [DM94]. Preference structures tend to be transitive, *i.e.* if $o_a \succeq o_b$ and $o_b \succeq o_c$ then $o_a \succeq o_c$. However, some decision scenarios might allow for $o_c \succeq o_a$. It is very difficult to construct a single consistent numeric utility function that successfully represents an intransitive set of preferences. Graphical models can also reveal independence in a preference structure. Bacchus and Grove explored a graphical representation of utilities with the weaker form of additive independence, called *conditional additive independence* [BG95]. Their work helps facilitate the decomposition of a utility function into somewhat independent components.

Boutilier *et al* created a network to capture and represent conditional preferential indepen-

dence [Bou01]. A *conditional preference network* (or CP-network) creates a node for every attribute X_i. For every attribute X_i, the user must identify a set of parent attributes whose values will influence the user's preference for the value of X_i. Each node, n_i, has an associated table describing how the parents' values will affect the preference for the value of X_i. With a set of initial conditional preference information, a CP-network can be used to rapidly decide which of two outcomes dominates the other or if there is an insufficient amount of information to determine the dominant outcome. In the case of the latter situation, the CP-network will identify an outcome whose preference information should be elicited from the user.

Conen and Sandholm make use of graphical structures to store preference information for a combinatorial auction setting [CS01b, CS01a]. They build an *augmented order graph* which is composed of a node for every possible bidder and bundle (*i.e.* a subset of the items up for auction). A feasible allocation (an outcome) is a set of nodes from the augmented order graph where no two nodes represent the same agent and no two nodes have bundles containing the same item. Algorithms use the augmented order graph to determine the set of Pareto-optimal allocations. An allocation A is Pareto efficient if there is no other allocation B where $v_i(A_i) \geq v_i(B_i)$ for each bidder i and $v_j(A_j) > v_j(B_j)$ for at least one bidder j (here A_i represents the bundle allocated to agent i by A). The graph can also be used to collect welfare-maximizing allocations, where the welfare-maximizing allocation A is one such that $\sum_{i=1}^{n} v_i(A_i)$ is maximized among all other allocations. Value, order, and rank information can then be elicited from the user from this computed set of allocations to reduce the set of candidate allocations to determine the optimal solution.

7 Picking the Next Query

The potential queries that an elicitor can present to a user have already been discussed. They were designed to reveal relevant information about the preferences of the user. However, elicitation should be an efficient process. Elicitors should try to collect the largest amount of preference information with the smallest number of questions. Therefore, elicitors should chose the next question with consideration and planning.

7.1 Metric Techniques

The most intuitive technique is to create a metric that evaluates the usefulness of potential questions. Various research projects have adopted a *value of information* metric which measures the expected improvement in preference information with the answer to a given query. The value of information function takes different forms for different decision support systems. For a possible query, Chajewska *et al* define an average of new expected utilities computed with each possible answer to the query factored into the utility function [CKP00]. The new utility values are also weighted by the likelihood of a particular answer from the user which is specified by a statistical model of the user. This average minus the current expected utility connotates their value of information function. Unfortunately, this function will only compute a *myopic* value of information, *i.e.* inconsiderate to the consequences with regard to future queries. The full

value of information is an intractable problem, requiring the look-ahead consideration of all possible future combinations of questions and answers. [Bou02] extends this work by building a partially-observable Markov decision process (or POMDP) model of the elicitation process to improve on the myopic scope of the value of information function in [CKP00].

Boutilier *et al* adopt a similar metric technique for their decision scenarios [Bou01]. For a given decision scenario with potential actions a_i, they examine the amount of *regret* a given action might receive. Factoring uncertainty into their model, the expected value of a given action a_i is defined as

$$EV(a_i, w) = \sum_{o \in O} Pr_i(o)u(o, w)$$

for a given a distribution of tradeoff weights w for the utility function. Define a^*_w as the action with the highest expected value, *i.e.* a^*_w is currently the best action to take for this decision problem. The regret of a_i with respect to w is defined as,

$$R(a_i, w) = EV(a^*_w, w) - EV(a_i, w)$$

which is a measurement for the loss incurred by performing a_i instead of the optimal action a^*_w. UCP-networks are used to form a set of linear constraints for the possible tradeoff weight values. These constraints define a subset of the distributions over outcomes O denoted as C.

For a given C, the *maximum regret* of action a_i with respect to C is defined as

$$MR(a_i, C) = max_{w \in C} R(a_i, w)$$

The set of constraints are then modified using queries with a finite number of responses in order to minimize the maximum regret value for C. The system submits to the user the query with the smallest maximum regret value. The answer to this query is then used to revise C. [WB03] improved on this work by removing the assumption that queries had a finite number of answers and fine tuning the regret-based query selection process.

Ha and Haddawy use a metric in their work called the *rank correlation coefficient* [HH97]. They needed a technique to measure the difference in the importance rankings of attributes between two different strategies. Considering permutations of the form $a = a_1, a_2, \ldots, a_n$ and $b = b_1, b_2, \ldots, b_n$, the rank correlation coefficient is defined as

$$\rho(a, b) = 1 - \frac{6 \sum i = 1^n (a_i - b_i)^2}{n^3 - n}$$

where the range of ρ is $-1 \leq \rho(a, b) \leq 1$. A value of 1 corresponds to two permutations being identical, while a value of -1 corresponds to complete reversal. The rank correlation coefficient was used to identify two strategies which differed the most from one another in terms of their rankings. The system then elicited the user for the proper information so that it could merge the attitudes of these two strategies into a single new strategy, thus reducing the number of candidate strategies for the user's consideration.

Value of information functions provide elicitors with an important tool for determining the next query to be posed to the user. However, the navigation assistant is trying to identify user

preferences without explicitly querying the user. Value of information functions appear to be of little use when omitting direct questioning of the user. However, the navigation assistant attempts to provide a degree of automation to assist with the exploration process. Boutilier's concept of regret may provide a tool for evaluating possible forms of automation within the visualization. Certain automated actions may have higher expected regret values (*e.g.* immediately taking control away from the user to bring an element to his attention) than do others (*e.g.* determining the order to visit potential points of interest for a user-requested automated tour). The primary mechanism for querying the user's interest in the navigation assistant is the presentation of candidate elements of interest. Using a value of information function to determine which elements would make good candidates is a type of exemplification discussed in the following section.

7.2 Providing Examples

A major disadvantage of valuation techniques are that they force the user to consider preferences via a fine granularity. The user must decide on exact numeric values that will accurately reflect his preferences. Coarse granularity considerations are often easier for a user to describe but might not always successfully generate the optimal outcome. An overall (or coarse) description can be provided by a user to create a subset of potential solutions. Then the system can start providing candidate solution examples to the user. Modification to the user model is based on the user responses to the candidates.

Providing example solutions to, or exampling, the user is perhaps the most intuitive way to elicit preference information. Users have the ability to rapidly and accurately return feedback about candidate solutions. They can inform the system about what attributes they like or dislike about a particular candidate. Furthermore, providing examples can alleviate possible interaction difficulties between the user and the decision support system. It is sometimes hard for a user to articulate preferences in a language that a decision support system can also comprehend. Most exampling systems operate using constraint solving techniques to determine candidate solutions.

O'Sullivan *et al* address exampling by considering it as an interactive constraint acquisition problem [OFO01]. They construct a *hypothesis space* for a given decision problem where user preferences are defined as constraints in their system. The hypothesis space's structure allows efficient pruning of potential outcomes based on the user's responses to suggested solutions. This structure also aids with the choice of the next example to suggest to the user.

The Automated Travel Assistant uses constraints to determine the appropriate solutions to present a user [LHL97]. Linden *et al* keeps the number of candidate solutions presented to the user small, only five. Therefore, when an undeveloped set of constraints produces a large candidate set, ATA needed to implement guidelines for effective solution selection. ATA always excluded dominated solutions from the candidate set. It prefers to present significantly different solutions by finding the candidates that minimize the error function but maximize the sum of the differences of attribute values. A follow-up guideline was to suggest extrema solutions because such solutions tend to optimize one attribute and tend to elicit more information about

the relative weighting of user preferences. This also allows ATA to potentially inform the user of the full range of solutions that meet the current set of preferences.

Exampling provides two types of mechanisms to allow elicitors to acquire additional preference information. The first is *tweaking*. The *FindMe* system allows users to explicitly define preference values via an interface for a set of attributes [BHY97]. *FindMe* then attempts to find solutions that are similarly based on the current set of preferences. Once presented with these candidates, users can then "tweak" their preference values to narrow their search until an optimal target solution has been located.

Instead of tweaking, other systems implement *example* or *candidate critiquing* [LHL97, SL01, TF99, PFT03]. These systems acknowledge that users are very efficient at recognizing violations of their preferences. Example critiquing can also help users discover new hidden preferences that were previously unknown to the user. This is important because users often have a difficult time with detailed refinement about their preferences. Users have a much easier time describing general, fundamental preferences. Example critiquing provides a mechanism that allows a system to determine the small quantitative differences among a user's values for similar solutions.

Most exampling systems depend on explicit user input for revising the set of candidate solutions. Users must either adjust some interface feature (tweaking) or describe the merits or flaws of a solution (candidate critiquing). Our navigation assistant is trying to reduce explicit interaction on the user's part. The navigation assistant already implements a form of exampling without any incremental revision of the candidate solution set, thus the exampling techniques described above could provide a great deal of benefit to the navigation assistant. Another important aspect of exampling systems is that they determine which solutions from a candidate set to display. Often, the solutions are displayed in a single view, side by side. This technique allows users to more easily make comparisons between candidate solutions and determine their true preferences. However, such displays can remove a solution from its meaningful context, potentially misleading the user about his value of a solution. The navigation assistant aims to avoid this by leaving candidates in context and smoothly moving the user through a visualization to the appropriate viewing locations. However, it is unclear how the efficiency of exampling is affected by how candidates are displayed to the user.

8 Conclusions

The navigation assistant in [DH02] aims to assist users with the exploration of a large data set. A major function of this system is to identify interesting data elements for the user and then assist the user with navigating to such elements. We hope to improve on the navigation assistant by allowing the implicit identification of elements of interest by observing user behavior within a visualization. Preference elicitation techniques attempt to acquire information about users' interests in an efficient and productive manner. It is possible for many ideas in preference elicitation to improve our navigation assistant.

The way many elicitation systems represent user preferences can easily be applied to a multidimensional dataset. Preference and utility independence may enable a system to bet-

ter understand how a user's behavior reflects his interests with particular groups of attributes. Common preference queries can provide helpful insight about how best to interpret user behavior to collect accurate preference information. Exampling is very similar to the current non-incremental implementation of the navigation assistant. However, most other exemplification elicitation systems require explicit response from the user in order to support incremental elicitation of the user's preferences.

From this survey of preference elicitation techniques, we hope to identify techniques and ideas that will potentially improve the effectiveness of our existing navigation assistant. We hope that these preference elicitation concepts will help with the identification of user interests while simultaneously facilitating the exploration process.

References

[BBGP97] Craig Boutilier, Ronen Brafman, Chris Geib, and David Poole. A constraint-based approach to preference elicitation and decision making. In *AAAI Spring Symposium on Qualitative Decision Theory*, pages 19–28, Stanford, CA, 1997.

[BF95] Daniel P. Boulet and Niall M. Fraser. Improving preference elicitation for decision support systems. In *Intelligent Systems for the 21st Century, Proc. of the 1995 International Conference on Systems, Man and Cybernetics*, Vancouver, BC, 1995.

[BG95] Fahiem Bacchus and Adam J. Grove. Graphical models for preferences and utility. In *Uncertainty in Artificial Intelligence (UAI-95)*, pages 3–10, 1995.

[BHY97] R. Burke, K. Hammond, and R. Young. The findme approach to assisted browsing. *IEEE Expert*, 12(4):32–40, 1997.

[Bou01] Craig Boutilier. Ucp-networks: A directed graphical representation of conditional utilities. In *Proceedings of the Seventeenth Annual Conference on Uncertainty in Artificial Intelligence (UAI-01)*, pages 690–697, Seattle, WA, 2001.

[Bou02] Craig Boutilier. A pomdp formulation of preference elicitation. In *Eighteenth National Conference on Artificial Intelligence (AAAI-2002)*, pages 359–2002, Edmonton, AB, 2002.

[CGNS98] Urszula Chajewska, L. Getoor, J. Norman, and Y. Shahar. Utility elicitation as a classification problem. In *Fourteenth Conference of Uncertainty in Artificial Intelligence (UAI'98)*, pages 79–88, San Francisco, 1998. Morgan Kaufmann Publishers.

[CK00] Urszula Chajewska and Daphne Koller. Utilities as random variables: Density estimation and structure discovery. In *Sixteenth Conference of Uncertainty in Artificial Intelligence 16 (UAI'00)*, pages 63–71, 2000.

[CKP00] Urszula Chajewska, Daphne Koller, and Robald Parr. Making rational decisions using adapative utility elicitation. In *Seventeeth National Conference on Artificial Intelligence (AAAI'00)*, pages 363–369, 2000.

[CS01a] Wolfram Conen and Tumoas Sandholm. Minimal preference elicitation in combinatorial auctions. In *International Joint Conference on Artificial Intelligence (IJCAI'01), Workshop on Economic Agents, Models, and Mechanisms*, Seattle, WA, August 2001.

[CS01b] Wolfram Conen and Tumoas Sandholm. Preference elicitation in combinatorial auctions [extended abstract]. In *ACM Conference on Electronic Commerce*, Tampa, FL, October 2001.

[DH02] Brent M. Dennis and Christopher G. Healey. Assisted navigation in large information spaces. In *Proceedings of IEEE Visualization '02*, Bostom, MA, 2002.

[DM94] Mitali De and Darren B Meister. Graphical forms for preference representation: Considerations for practical application. In *1994 IEEE SMC Conference*, San Antonio, TX, 1994.

[Fre86] Simon French. *Decision Theory: an introduction to the mathematics of rationality*. Ellis Horwood Limited, 1986.

[HH97] V. Ha and P. Haddawy. Problem-focused incremental elicitation of multi-attribute utility models. In *Proceedings of the Thirteenth Conference on Uncertainty in Artificial Intelligence*, pages 215–222, August 1997.

[HH98] V. Ha and P. Haddawy. Towards case-based preference elicitation: Similarity measures on preference structures. In *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*, pages 193–201, July 1998.

[HH99] V. Ha and P. Haddawy. A hybrid approach to reasoning with partial preference models. In *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*, pages 263–270, August 1999.

[HHR+03] Peter Haddawy, Vu Ha, Angelo Restificar, Benjamin Geisler, and John Miyamoto. Preference elicitation via theory refinement. *Journal Machine Learning Research*, 4:317–337, 2003.

[HS02] Benoit Hudson and Tuomas Sandholm. Effectiveness of preference elicitation in combinatorial auctions. In *Agent-Mediated Electronic Conmmerce (AMEC) workshop at AAMAS-02*, Bologna, Italy, 2002.

[KR76] Ralph L. Keeney and Howard Raiffa. *Decisions with Multiple Objectives*. Cambidge University Press, 1976.

[LHL97] Greg Linden, Steve Hanks, and Neal Lesh. Interactive assessment of user preference models: The automated travel assistant. In *Proceedings of User Modeling '97*, 1997.

[NM47] John Von Neumann and Oskar Morgenstern. *Theory of Games and Economic Behavior*. Princeton University Press, 1947.

[OFO01] Barry O'Sullivan, Eugene C. Freuder, and Sarah O'Connell. Interactive constraint acquisition. In *Proceedings of Workshop on User-Interaction in Constraint Processing at the CP-2001*, 2001.

[PFT03] Pearl Pu, Boi Faltings, and Marc Torrens. User-involved preference elicitation. In *Working Notes of the Workshop on Configuration. Eighteenth International Joint Conference on Artificial Intelligence (IJCAI-2003)*, Acapulco, Mexico, August 2003.

[SL01] Sybil Shearin and Henry Lieberman. Intelligent profiling by example. In *Proceedings of the Conference of Intelligent User Interfaces, IUI'01*. ACM Press, 2001.

[TF99] Marc Torrens and Boi Faltings. Smartclients: Constraint satisfaction as a paradigm for scaleable intelligent information systems. In *AAAI Workshop on Artificial Intelligence for Electronic Commerce*, volume Technical Report WS-99-01, pages 10–15. AAAI, AAAI Press, July 1999.

[WB03] Tianhan Wang and Craig Boutilier. Incremental utility elicitation with minimax regret decision criterion. In *International Joint Conference on Artificial Intelligence (IJCAI'03)*, 2003.