Q-Eval: Evaluating Multiple Attribute Items Using Queries

Vijay S. Iyengar
vsi@us.ibm.com

Jon Lee
jonlee@us.ibm.com

Murray Campbell
mcam@us.ibm.com

IBM Research
T.J. Watson Research Center
Yorktown Heights, NY 10598 USA

ABSTRACT

The task of evaluating and ranking items with multiple-attributes appears in many guises in commerce. Examples include evaluating responses to a request for quotes (RFQ) for some item and comparison shopping for an item within one or more catalogs. This task is straightforward if the value of the item can be explicitly specified by the evaluator as a function of the attribute values. However, a typical evaluator may not be able to provide the value function in explicit form. In contrast, it is intuitive for them to compare, say, two items and pick the preferable one based on all of the relevant attributes. In this paper we present a method, Q-Eval, that queries the evaluator with selected pairs of items and uses the responses to build a preference model for the evaluator. This model is then used to rank the items in order of the inferred preference. The evaluator can then pick the winning item or items by considering only the top few items in this ranked list. This should result in significant productivity improvement for the evaluator when the number of items to choose from is large. Our algorithm is novel in the way it attempts to derive a stable preference model with only a small number of user queries. This paper describes the algorithm and presents experimental results with real-life data to validate the approach.

General Terms

Multiple-attributes items, bid evaluations, comparison shopping, utility functions, preference elicitation

1. INTRODUCTION

Evaluation and selection from a set of items with multiple-attributes is a task performed in various aspects of commerce. For example, bids received as responses to a request for quotes (RFQ) need to be evaluated and the winning bid or bids selected. For illustration, consider bids received in response to a request for ten laser printers for an office. To evaluate the bids[1], one may consider attributes like price, brand and model, print speed, print resolution, cost per page, paper capacity, warranty, and maintenance service terms. The evaluator[2] would then compare the attribute values for all the bids received to choose the winning bid or bids.

Examples of evaluating multiple-attribute items also exist in the consumer space. A consumer shopping for a particular item is confronted with choices not only in terms of products from various manufacturers but also in terms of various channels for purchasing these products. In addition to the product attributes, the consumer may also consider attributes of the channel like return policy and customer satisfaction rating of the channel by some independent agency. As in the bid evaluation case, the consumer compares the items, based on all the important attribute values, to pick the one to purchase.

In both of these examples the task of evaluation becomes tedious if there are many items that have to be compared. The number of attributes to be considered also adds to the complexity of this task. In this paper we describe a new algorithmic approach (Q-Eval) to this task that selects pairs of items in an iterative process for comparison by the user. The comparison outcomes are used to build a model of the user's preferences based on the item attributes. The items are then ranked in order of preference as suggested by the derived model. A novel aspect of our algorithm is the method by which pairs of items are selected for comparison to try and derive a stable model within a small number of iterations (i.e., with few pairwise comparisons). If the user preference model is consistent with the evaluator's decision making then we can achieve significant productivity gains by having the evaluator pick the winning item from the top few items in our ranked list without ever having examined all of the items.

2. BACKGROUND

The goal of this work falls into the broad area of decision analysis and is related more specifically to eliciting preferences and multi-attribute utility theory (MAUT) [15, 4]. Making a decision, such as ranking multi-attribute items, involves trade-offs across the various attributes. Utility functions can be used to capture a decision maker's attitude to-

[1]The RFQ might have specified constraints for some attributes in terms of allowed values (e.g., minimum print resolution). The bids satisfy any such constraints.

[2]We will use the terms evaluator or user to refer to the person doing the evaluation.

EC'01, October 14-17, 2001, Tampa, Florida, USA.

wards each attribute (e.g., price of item) [4, Chapter 13], and these can be combined to determine a utility score or value for an item. A decision maker should make choices consistent with the maximization of item value if they behave according to some intuitive behavioral axioms [4, Chapter 14].

Our work assumes that there is no uncertainty during the decision making. For simplicity, most MAUT techniques assume an additive model to represent the evaluator's value of an item [4, Chapter 15]. In this model, the value of an item can be expressed as

$$value = \sum_{i=1}^{d} f_i(x_i) \times w_i \qquad w_i \geq 0 \tag{1}$$

where x_i is the value of the attribute i for the item, w_i is a weight indicating the importance of the attribute i to the buyer, and f_i is the utility function for the ith attribute.

The simplest version of this model uses linear utility functions f_i, leading to a linear function for the buyer's value in terms of the attribute values. However, this simple model does not satisfy the principle of diminishing marginal utility [18]. Consider an attribute for which larger values are preferable, e.g., print speed for a laser printer. According to this principle, the additional utility to a buyer due to a constant increase in print speed is less at higher print speeds.

Other functions have been proposed for f_i. The issue with the diminishing marginal utility principle can be addressed by using the Cobb-Douglas form [18] given below for the value function:

$$value = \prod_{i=1}^{d} (x_i)^{w_i} \qquad w_i \geq 0, \tag{2}$$

where w_i reflects the importance of attribute i. Taking logarithms on both sides of Equation 2 leads us to

$$log(value) = \sum_{i=1}^{d} log(x_i) \times w_i \qquad w_i \geq 0, \tag{3}$$

which is in the form of Equation 1 with f_i replaced by the *log* function.

In this paper, we will choose one of these two additive models using either the linear or the logarithmic utility functions for each attribute. Determining a buyer's preferences will then involve discovering the attribute weights that best model these preferences.

Various methods have been proposed to assess the weights for additive models [4, Chapter 15]. For example, the swing weighting approach requires the decision maker to compare hypothetical items with various combinations of best and worst values for each attribute [4, Chapter 15]. The analytic hierarchy process (AHP) [19] is another approach to discovering the attribute weights by using an attribute hierarchy. Weights for the attributes are derived by having the user compare attributes sharing a parent in the hierarchy.

An alternative approach commonly used in marketing to measure consumer preferences for a multi-attribute product is conjoint analysis [10, 11, 16]. Attribute values are usually discretized and items with all the combinations of discrete attribute levels are presented to the user for ranking. The rank of a product in the complete ordering is used as its value. An additive model is then fitted using statistical methods like regression analysis. Extensions to this basic method have also been proposed that address the issue of having to consider too many cases if there are many attributes and / or levels. One such extension, adaptive conjoint analysis, starts out by getting a rough ordering of attribute importance from the user [12, 13, 11]. In a subsequent step, customized pairs are presented for comparison by the user. The heuristics used to choose the pairs try to pick products that are roughly equal in value while balancing the usage of various attributes and levels.

A recent project, Multi-Attribute Resource Intermediary (MARI), describes a system to capture user preferences with the goal of improving online marketplaces [21]. MARI also uses an additive model for the value of an item. In MARI, utility functions for the attributes are first visually selected by the user. The user is asked to specify ranges of permissible values for each attribute. These ranges are used to derive the weights w_i for each attribute using the intuition that attributes with tighter constraints are the more important ones (i.e., have higher weights w_i). This tight coupling between weights and attribute constraints can be restrictive.

Another closely related implementation is provided by Active Buyer's Guide^{TM} at www.ActiveBuyersGuide.com to help consumers shop for items. This system first queries the consumer to specify the importance of the various item attributes. Then the system iteratively queries the user to pick the preferred item from a pair of synthetically generated items. After some rounds of this iterative process, a ranked list of items is presented to the consumer.

Our approach improves over these earlier methods in the following ways. In our approach we do not expect the user to specify relative importance of various attributes. Also, presenting synthetic items to compare may be misleading to a user regarding which items are available. Hence, in our approach only available items are presented for comparison. Moreover, our algorithm uses well defined criteria to systematically refine the model so that an accurate ranked list can be generated with a small number of user queries. In contrast to conjoint analysis, in our work the item comparisons are used to generate constraints on the values (which are unknown since item ranks are not used as values). Using ranks as values assumes that differences in values are uniformly related to differences in ranks. The next section describes our algorithm.

3. THE Q-EVAL METHOD

3.1 Overview

Discovering the utility function f_i is not a part of our method and we expect it to have been chosen by the user or based on the domain. This implies that the mapping from x_i to $f_i(x_i)$ is known, and so we can rewrite Equation 1 in the following form

$$value = \sum_{i=1}^{d} y_i \times w_i \qquad w_i \geq 0, \tag{4}$$

where $y_i = f_i(x_i)$ represents the utility for attribute i. In our experiments we will consider some specific examples of the utility functions f_i. Two specific types of utility functions of interest are the linear function and the logarithmic function discussed earlier.

A high-level description of the Q-Eval algorithm is presented in Figure 1. The input to the algorithm is the list

Method Q-Eval (Input: Items for evaluation with multiple attributes,
Output: Ranked list of items,
Output: Attribute weights)
1 Allow user to select the set of relevant attributes
(Step 1 is optional, default: all attributes are relevant)
2 **For** each relevant numeric attribute choose
polarity *(whether increasing or decreasing values are preferable)*
utility function *(could also be done uniformly for all attributes)*
3 **For** each relevant categorical attribute
Have user map the categorical values to a numerical scale *(user indicates relative preference for categorical values)*
4 Initialize attribute weights
5 **For** each iteration
6 Select a pair of items P = (a,b)
7 Query user to choose preferred item in pair P
8 Recompute attribute weights using user response
9 Present ranked list of items to user using final attribute weights
end Q-Eval

Figure 1: Description of the Q-Eval method of evaluation and ranking by queries (comments are *italicized*)

of items with the attribute values. The output of the algorithm is a ranked list of items sorted by the value to the evaluator as inferred by the algorithm. The inferred values of the items based on Equation 4 are also provided with the ranked list. The inferred values are useful to determine relative positioning of the items based on the preference model. Weights as defined in Equation 4 are also available as output when the algorithm finishes. However, it is unclear how intuitive these weights are to the evaluator.

Step 1 of Q-Eval is optional and allows the user to select the set of relevant attributes. If some attributes are definitely and obviously irrelevant to the user then specifying this at the beginning can reduce the dimensionality of the problem which in turn can lead to better and faster solutions. We will drop the *relevant* qualifier for attributes from now on and assume that only relevant attributes being considered. Step 2 specifies a polarity indicating whether increasing or decreasing values are preferable for each numeric attribute. The utility function could also be chosen for each attribute. However, in our experiments we focus on using either linear or logarithmic utility functions uniformly for all attributes.

Some of the attributes are categorical in nature (e.g., printer brand). Such attribute values are mapped to a numerical scale by the user in Step 3 of Q-Eval. This mapping allows the user to indicate relative preference for various values for these attributes and facilitate their usage in Equation 4.

Steps 4-8 in Figure 1 represent the iterative process of querying and refining the user preference model. A selected pair of items P is presented to the user. Attribute values for both items in P are displayed and the user is queried to choose the preferred item in P. The user response is used to refine the preference model. The refined model is used to rank all the items in the list in the order of preference as indicated by the model. At the end of this iterative query process the final ranked list of items is presented to the user. The next subsection describes the algorithm in detail.

3.2 Algorithmic details

Consider again the model for the total value of an item in Equation 4. The value of an item is represented as a linear function of the attribute weights. Since we are interested only in the relative values and not the absolute values of items, the weights can be normalized using Equation 5, without any loss of generality.

$$\sum_{i=1}^{d} w_i = 1 \qquad w_i \geq 0 \tag{5}$$

This effectively reduces the dimensionality of the weight space by one to $(d-1)$ by expressing one of the weights, say w_d, as a function of the rest

$$w_d = 1 - \sum_{i=1}^{d-1} w_i. \tag{6}$$

Most of the operations in this subsection will be in the $(d-1)$ dimensional weight space with the understanding that the remaining weight can be computed using Equation 6.

The Q-Eval algorithm keeps track of the permissible region R in the $(d-1)$ dimensional weight space (simply called weight space from now on). The region R is represented by the linear constraints and the corresponding half-spaces that define it. Using Equations 5 and 6, the region R can be initially (prior to any queries) defined by Equation 7.

$$\begin{aligned} &0 \leq w_i \leq 1 \quad i = 1, \ldots, d-1 \\ &\sum_{i=1}^{d-1} w_i \leq 1 \end{aligned} \tag{7}$$

The permissible region, as the name implies, represents the possible weights that are consistent with the information obtained from the user via queries. Intuitively, we want to have a small region implying that we have narrowed down the possibilities for the weights. Q-Eval attempts to do this by the choice of item pairs that are presented to the user for ranking as described next. Consider a pair of items $P = (a, b)$. Let y_i^a and y_i^b represent the utility for the ith attribute of a and b, respectively. If the user specifies that item a is preferred over item b when the pair P is presented for comparison, we can infer that item a has higher value as expressed by Equation 8 below.

$$\sum_{i=1}^{d} w_i \times y_i^a \geq \sum_{i=1}^{d} w_i \times y_i^b \tag{8}$$

Using Equation 6 to eliminate w_d from Equation 8, we can derive Equation 9 below.

$$\sum_{i=1}^{d-1} w_i \times [(y_i^a - y_d^a) - (y_i^b - y_d^b)] \geq (y_d^b - y_d^a) \tag{9}$$

Equation 9 specifies another constraint in terms of a half-space that the permissible region must satisfy. Ideally, we would like to choose the pair P so that this additional constraint maximally shrinks the permissible region R. However, without querying the user, we do not know a priori

which of the two items in the pair P the user would prefer. Hence, choosing the item pair P that comes close to bisecting the current permissible region makes intuitive sense. However, the shape of the region is another factor to be considered when choosing the item pair P. Constraining the region to have a small range of values in each dimension is also an intuitively desirable property.

Method Q-Eval: Details of Steps 4-9
4a Initialize permissible region R using Equation 7
4b Compute center C of the region R
4c Initialize attribute weights
5 **For** each iteration
6a Compute the normal distance L of center C to the hyperplane H corresponding to each item pair P
6b Sort item pairs in order of increasing distance L
6c For each of the k hyperplanes H closest to C
Compute the volumes of the two parts of R split by H
6d Choose the hyperplane and corresponding item pair P with the best split of R
7 Query user to choose preferred item in pair P
8a Add linear constraint for half-space chosen to region R
8b Recompute center C of region R
8c Recompute attribute weights
9a Compute ranked list of items based on computed attribute weights
9b Display ranked list and computed item values
end Q-Eval: Details

Figure 2: Details of the Q-Eval algorithm

The key steps of Q-Eval are described in greater detail in Figure 2. Table 1 has the bids for a small example that will be used to illustrate our algorithm. This example has three bids (B1, B2, and B3) with three attributes (X0, X1, and X2). The attribute values (after normalization) for the three bids are given in Table 1. The table also gives the evaluator's preferences in terms of attribute weights (W0, W1, and W2) and the corresponding *true* values for the three bids. Weight W2 is dropped using Equation 6 to get the two dimensional weight space (W0, W1). In step 4a of Q-Eval, the permissible region R in the weight space is initialized using the constraints in Equation 7. Step 4b computes the center C of the region R. The purpose of computing the center C is twofold. The Q-Eval algorithm has to pick a set of attribute weights from the permissible region R which can then be used to rank the items. This is done by using the coordinates of the center to pick weights for the $(d-1)$ attributes used in the weight space and then using Equation 6 to calculate the dth attribute weight. The second use of the center will be described later in Step 6c. For the example, the initial region R with its center is shown shaded in Figure 3 (top).

We use the notion of *prime analytic center* (see [3, 20]) in all the experiments in this paper. Computing the prime analytic center requires that we first determine the irredundant set of constraints after each query response using linear programming. The center is then computed by maximizing the sum of the logarithms of the normal distances of an interior point to the irredundant hyperplanes defining the region. In our implementation we start from an interior point solution, obtained by solving a linear program, and then use a steepest ascent method with an exact line search to calculate the prime analytic center. This approach seems adequate for the low dimensionality of the problems encountered in our applications. If slow convergence seems to be an issue in higher dimensions, a Newton or Quasi-Newton method should provide better search directions and faster convergence [17]. Step 4c computes the initial set of weights from the coordinates of the center as discussed above.

Bids	X0 (W0=0.7)	X1 (W1=0.2)	X2 (W2=0.1)	True value
B1	20	10	10	17
B2	10	20	10	12
B3	5	5	35	8

Table 1: Example with three attributes

Steps 6a-6d choose the item pair P to be presented in the query to the user. As explained earlier, this choice is made by determining which pair would best shrink the current region R. The total number of item pairs to consider can be quite large. Two heuristics are used to address this problem. First, we will consider only those item pairs whose corresponding hyperplanes are close to the center of the region R. Intuitively, these are more likely to be the better candidates for bisecting R. This is accomplished in steps 6a and 6b by sorting the item pairs by increasing normal distance from the center of region R. In step 6c, only the closest k hyperplanes are evaluated by computing the volumes of the two parts of R split by each hyperplane. This heuristic is analogous to the choice of items with roughly equal estimated value in adaptive conjoint analysis. The second heuristic is to restrict the item pairs being considered in step 6c to those that contain only the top few items in the current ranking. In our experiments, only item pairs containing the top 30 items based on the current weights are considered in step 6c. This also has the added advantage of presenting items closer to the user's preferences for comparison in the query. Figure 3 (top) shows the three hyperplanes (lines in 2 dimensions) corresponding to the three possible item pairs in the example. The hyperplane corresponding to the item pair (B1, B2) is the closest to the initial center of region R (normal distance zero).

The volumes computed in step 6c could be the exact volumes of the split up parts of R or some approximation of them. Calculating the exact volume from the list of inequalities has been shown to be hard [6]. Constraints on the elapsed time between the user response and the next query in Q-Eval force us to consider simple approximations to the volume computation.

Our approximate volume calculations are based on the tightest axis-orthogonal bounding box that encloses the region. One approximation used is the volume of this bounding box. Our initial experiments indicated that the shape of the region was also important when the prime analytic center was used. This led to the other approximation which is simply the maximum length (normalized) of the bounding box over all dimensions. Intuitively, using this approximation will prevent the region from extending too much along any axis. The length of the bounding box in each dimension

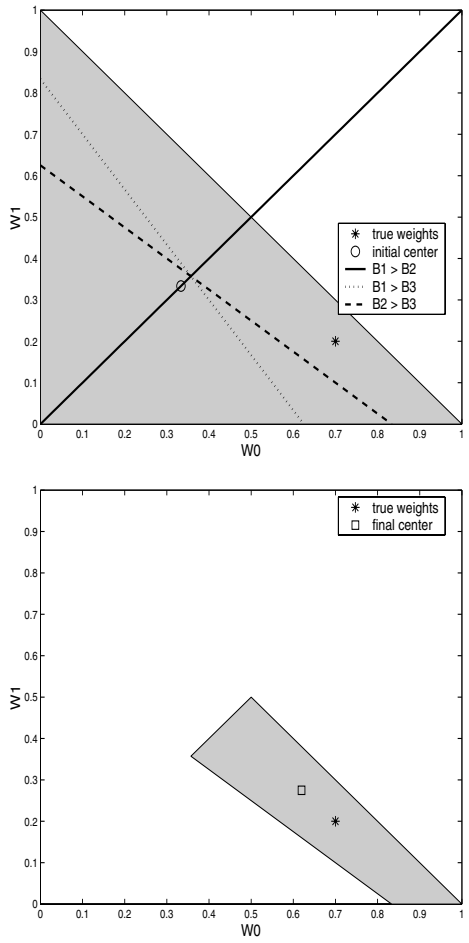

Figure 3: Initial (top) and final (bottom) permissible regions for example in Table 1

can be determined for either of these two approximations by solving two linear programs. Experiments presented in the next section indicate that these simple approximations are quite effective.

In step 6d, we choose the item pair P with the hyperplane that best shrinks the feasible region amongst those considered. The chosen item pair P is presented to the user as a query in Step 7. The user's response generates a new constraint as described in Equation 9 that is added to further cut the feasible region. Steps 8b and 8c compute the center of the new feasible region and the corresponding attribute weights. For the example, the hyperplane corresponding to item pair (B1, B2) would be chosen as having the best split of the region R in Figure 3 (top).

In each iteration of steps 5-8, a new query is presented to the user and the response used to further shrink the permissible region of weights. In practice, there will be an upper bound on how many queries the user can be subjected to. Conflicting constraints that result in an empty permissible region can be detected and will cause the iterative process to terminate with the appropriate signal to the user. An empty permissible region could indicate an error in at least one of the user responses or could be due to the inadequacy of the value model used in our formulation. Some notion of convergence can be used to determine if more queries are needed to refine the permissible region further. Clearly, we are done if none of the hyperplanes corresponding to the item pairs can further refine the region. A weaker notion of convergence could be based on changes in the location of the center during the iterative process. The final permissible region R for the example is shown shaded Figure 3 (bottom) after all three possible item pair queries have been answered by the user. The center corresponding to the user's true weights and the computed center are also marked. Note that the region R cannot be shrunk any further since all possible pairwise comparisons have been done.

The final steps, 9a and 9b, compute the ranked list based on the weights computed from the final permissible region. These are presented with the computed values of the items which can be useful as an indication of the relative values of the items as inferred by the system. For the example, the pairwise queries are presented in the order (B1, B2), (B1, B3) and (B2, B3) using any of our volume approximations. The three items are correctly ordered after the response to the second query.

4. EXPERIMENTS

Experiments were performed by applying the Q-Eval method to some real datasets. Use of synthetic datasets was not pursued because the results of Q-Eval depend a great deal on the distributions for attribute values amongst competing items. For example, is it realistic for one item to stand out amongst the competing ones for any set of attribute weights?

For Q-Eval to be successful in actual applications two conditions have to be satisfied. First, the models used for utility and value in Q-Eval must fit the evaluator's notion of preferences. Second, Q-Eval must derive the parameters of the model using only a small number of queries. Validating both these conditions for success requires subjective experiments with actual evaluators. We have done a very limited number of experiments along these lines so far. In this paper we will focus on experiments that try to assess the success of the method using only the second condition discussed above. Hence, in our experiments we assumed certain value models and then assessed how the method performed in trying to recover these models.

4.1 Datasets

Two datasets were used in our experiments. The first dataset describes various digital cameras available in the market. This dataset was manually created by collating a list of products available from various manufacturers. Product features were obtained from the product specifications and a representative street price was used for each product. The digital camera dataset (referred to as the camera dataset) has a total of 88 products from 16 manufacturers. The following seven attributes were used for the products: brand name, price, resolution (pixels), type of storage media, optical zoom range, LCD screen size and weight.

The second dataset contains 83 inkjet printers with the following eight attributes: price, brand, weight, sheet capacity, maximum paper size, color resolution, color and b/w output speeds.

4.2 Experimental setup

Each experiment was done with one of these two datasets by choosing a utility model and some specific weights for attributes that represent some evaluator's preference. As mentioned earlier, either the linear or the logarithmic utility function was used uniformly for all numeric attributes in our experiments. Values for categorical attributes were mapped to a suitably normalized scale using some assumed user preferences. In each experiment we will compare the results achieved by the following variants of Q-Eval:

A1 Q-Eval using the maximum length (normalized) in any dimension of the tightest axis-orthogonal bounding box enclosing the permissible region.

A2 Q-Eval using the volume of the tightest axis-orthogonal bounding box enclosing the permissible region.

A3 Q-Eval with the hyperplane to cut the permissible region chosen randomly.

In algorithms A1 and A2 if two hyperplanes are tied with respect to the notion of volume used then the hyperplane closest to the center is chosen. The default condition for algorithms A1 and A2 is to restrict the item pairs considered for splitting the region to those that use only the top 30 items in the current ranking. The value of k in step 6c of Figure 2 is set to 100 for the algorithms A1 and A2.

These algorithms will be compared using various metrics. First, we will plot the reduction in the actual volume of the permissible region with increasing number of pairwise queries and responses. The exact volume was computed using the revised Lasserre's method (r-LAS) as implemented in the *Vinci* system and described in [2]. A smaller region implies fewer possibilities for the weights. Comparing the derived weights with the actual weights is not a good way to assess the quality of the results. Since attribute values can be correlated it is possible to achieve perfect ranking of the items with derived weights that are different from the actual weights. Instead, we will measure the achieved closeness to perfect ranking as a function of the number of queries. The rank of the top item inferred by an algorithm is used as a ranking metric to judge how quickly it rises to the top of the ranked list. Judging using only the top item does not take into account the situation that the top few items might be very close in value to the user. Hence, we will also use the number of actual top-L items in the top-L positions of the computed ranked list as another ranking metric. Specifically, we will use values of 3 and 5 for L to measure if the top-3 and top-5 items are being identified correctly.

Reduction in the volume of the permissible region by an algorithm is a good way to assess its robustness. One potential problem with the ranking metrics is that they can show good results for an algorithm because the center of the region is *correct* even though the region is still quite large. Hence, both volume and ranking metrics will be used to compare different algorithms.

4.3 Results

The experiments reported in the paper are listed in Table 2. Table 2 specifies the dataset, the utility function used uniformly over all the numeric attributes, the number of attributes considered (i.e., the dimensionality of the problem) and the actual weights. When using linear utility functions, we normalize the attribute values to have zero mean and standard deviation of one. For the logarithmic utility function the values are again normalized using the standard deviation to have a minimum value of one prior to taking logarithms.

In the first experiment, the evaluator is interested in a cheap and light camera with weights of 0.5 for the price and weight attributes. Figure 4 shows the reduction in the volume of the permissible region (semilog plot) for the variants of Q-Eval discussed earlier. The exact bisection of the volume with each query is useful as a reference (marked as *bisection*). The comparison with the exact bisection line indicates that algorithms A1 and A2 do quite well in shrinking the volume rapidly with the queries. The volume curve for algorithm A1 flattens out towards the end because of the heuristic that restricts the items pairs considered to include only the top 30 items. This impacts algorithm A1 (more than algorithm A2) because of its focus on the worst dimension of the bounding box which might be difficult to reduce with the restricted set of hyperplanes. The heuristic restricting the items has to be applied carefully so that we get the benefit of presenting relevant items to the user without prematurely losing the ability of shrink the permissible region. The performances of two random trials (T1, T2) of A3 are also shown in Figure 4. Trial T1 and T2 were chosen from a set of 5 random trials of A3 as the ones having the best and worst volume after all the 30 queries. Random selection of queries can perform poorly as the worst case shown indicates. The algorithms were also compared using the ranking metrics. The position of the top item achieved by each algorithm is shown (semilog plot) in Figure 5. Algorithm A2 is the quickest to identify the top item in just 9 queries while algorithm A1 takes 11 queries. The poorer results achieved by the same random trials T1 and T2 are also shown. The algorithms were also compared using the top-3 and top-5 metrics in Figure 6. Algorithm A2 gets the top-3 and four of the top-5 correct after just 6 queries but takes 21 queries to get all the top-5 items. Algorithm A1 gets all the top-3 and four of the top-5 after 10 queries and all the top-5 after 14 queries. The random trials T1 and T2 show significant variation in their performance using the ranking metrics.

The results for all experiments are summarized in Table 3. The performance of each algorithm is reported after 5, 10, 15 and 20 queries are completed. For algorithm A3 each entry gives the range of values achieved in the 5 trials by indicating the worst and the best values achieved for each metric. Note that no single trial of algorithm A3 would necessarily achieve all the best (or worst) values shown in Table 3 for an experiment. The volume metric reported in this table is calculated using Equation 10.

$$Volume\ metric = \log_2 \frac{Volume(algorithm)}{Volume(bisection)} \qquad (10)$$

This metric measures the gap between the volume of the permissible region achieved by the algorithm and the volume achieved by bisection using the logarithmic scale. For example, a value of -1 for this metric indicates that the algorithm outperformed bisection by achieving half the volume. Table 3 also reports results using the ranking metrics, namely, the position of the top item and the top-3 and top-5 metrics.

The dimensionality of the problem will clearly impact the

Experiment	Dataset used	Number of items	Utility function	Number of attributes considered	Non-zero attribute weights	
					Number	Details
1	cameras	88	linear	7	2	price: 0.5 weight: 0.5
2	cameras	88	linear	4	2	price: 0.5 weight: 0.5
3	cameras	88	logarithm	7	2	price: 0.5 weight: 0.5
4	printers	83	logarithm	8	3	price: 0.5 color speed: 0.25 color resolution: 0.25

Table 2: List of experiments

number of queries needed to get a good solution. If the user can identify some attributes as being irrelevant for the evaluation then the dimensionality of the problem can be reduced. Experiment 2 illustrates this by reducing the dimensionality of the first experiment by dropping three attributes: type of storage media, optical zoom range, and LCD screen size. Algorithms A1 and A2 both identify the top item after only 7 queries and take only 11 queries to get all the top-5 correct. Other results for this experiment are summarized in Table 3.

Experiment 3 repeats the first experiment except that the linear utility functions are replaced by logarithmic functions. The results are given in Table 3. Algorithms A1 and A2 perform well with the logarithmic utility functions. The volume metric indicates that both A1 and A2 come close to bisecting the volume after each query. The ranking metrics indicate that the top items are also identified by algorithms A1 and A2 after about 10 queries.

Experiment 4 uses the printers dataset with the logarithmic utility functions. The user is interested in a low cost color printer with high speed and resolution. This is modeled with 3 non-zero attribute weights as shown in Table 2.

Results are summarized in Table 3. Algorithm A1 does better than A2 in shrinking the volume of the permissible region. However algorithm A2 comes out ahead when considering the ranking metrics.

In our experiments, algorithms A1 and A2 do quite well in terms of the volume and the top-L metrics and also have manageable computational requirements. The erratic performance of the random algorithm A3 reinforces the intuition that queries have to be chosen carefully. In the next section, we discuss some possible extensions to these algorithms including some of our future directions.

5. DISCUSSION

The results using algorithms A1 and A2 suggest that the volume and the shape of the permissible region are important to identify the top items with a small number of queries. Other ways of quantifying the size and shape of the region need further exploration. We have done some preliminary experiments using the sum of the lengths of the permissible region's bounding box as an alternative way of characterizing its size and shape. Results achieved so far are comparable to those achieved by algorithms A1 and A2.

Various approximate volume computations have been proposed in the literature. Randomized algorithms have been proposed for approximating the volume of a convex body [7, 14]. The maximum volume inscribed ellipsoid is another approximation worth considering [23]. Affine transformations to the region have also been applied to improve the approximations [1]. However, it remains to be seen if the extra computational cost for such methods leads to better performance in terms of identifying the top items quickly.

There are various notions of a center for a convex region. The center of mass assuming that the mass is uniformly distributed over the region has intuitive appeal but it is expensive to compute. Initially, we considered using the vertex-barycenter, which is computed as the simple average of the coordinates of the vertices of the convex region. The vertex-barycenter can be computed by first enumerating the vertices of the region using packages like the CDD [8, 9]. However, since this is known to be a computationally intensive problem [5, 22] the use of the vertex-barycenter was abandoned.

In our experiments, the center was initially set so that all attributes were equally weighted. If the evaluator's bias is known it can be used to initialize the center and the first item pair can be chosen based solely on the distance from this biased center.

It is also interesting to consider extending the notion of comparing pairs of items to comparing say three items. Ease of use is the first issue to consider. Is it easier to do fewer rounds of comparing three items than doing more rounds of comparing item pairs? Extending the basic idea of reducing the permissible region volume when comparing item triples is straightforward. At any point in the algorithm the triple selected for the user query should have the smallest resulting volume assuming that any of the three could be chosen as the best by the user. However, there are many open issues relating to efficiency in handling triples.

There has been extensive research on utility functions. Interactions between attributes require more complicated models of utility. One extension to the additive model in Equation 1 adds explicit terms of the form $f_j(x_j) \times f_k(x_k) \times w_{jk}$ to the value of an item. Another way of allowing more complicated utility models for an attribute is by breaking up the attribute values into various ranges and representing them as new attributes. As long as the total number of additive terms is not too large these extensions can be handled by Q-Eval as a problem with higher dimensionality. Handling more complex utility functions and inconsistent responses from the user need further investigation.

6. CONCLUSIONS

In this paper we have presented a novel method of evaluating multi-attribute items using queries (Q-Eval). Our method is applicable if the user's preferences can be modeled by a reasonably general class of value functions. Pairs of items are chosen by Q-Eval for ranking by the user with the goal of extracting the user preference model using few queries. This is accomplished by keeping track of a permissible region in a weight space, where the weights represent the relative importance of the item attributes. Item pairs that best shrink this permissible region are presented for ranking by the user. Experimental results on real datasets were used to compare various algorithms with different computational requirements. These results suggest solutions that have reasonable computational requirements and performance as measured by our metrics. Our experiments also indicate interesting directions for future work.

Acknowledgments

We would like to thank Chung-Sheng Li for motivating the use of iterative refinement techniques in this domain. We would also like to acknowledge the helpful discussions with Alexei Zarovnyi, Jonathan Leland and Steven Gjerstad and the constructive comments from the referees.

7. REFERENCES

[1] S. Arnborg. Learning in prevision space. In *Proceedings of 1st International Symposium on Imprecise Probabilities and their Applications*, 1999.

[2] B. Bueler, A. Enge, and K. Fukuda. Exact volume computation for convex polytopes: A practical study. In *Polytopes - Combinatorics and Computation*. DMV-Seminars, Birkhauser Verlag, 1998. FTP site=ftp://ftp.ifor.math.ethz.ch/pub/fukuda/reports.

[3] R. Caron, H. Greenberg, and A. Holder. Analytic centers and repelling inequalities. CCM 142, Center for Computational Mathematics, Mathematics Department, University of Colorado at Denver, Denver, CO, 1999.

[4] R. Clemen. *Making Hard Decisions*. Duxbury Press, 1996.

[5] M. Dyer. The complexity of vertex enumeration methods. *Mathematics of Operations Research*, 8(3):381–402, 1983.

[6] M. Dyer and A. Frieze. On the complexity of computing the volume of a polyhedron. *Siam Journal of Computing*, 17(5):967–974, 1988.

[7] M. Dyer, A. Frieze, and R. Kannan. A random polynomial time algorithm for approximating the volume of convex bodies. *Journal of the ACM*, 38(1):1–17, 1991.

[8] K. Fukuda. Cdd - an implementation of the double description method. Technical Report Technical Report, DMA-EPFL, Institute for Operations Research, ETH Zentrum, Zurich, 1993. URL=http://www.ifor.math.ethz.ch/~fukuda/cdd_home/cdd.html.

[9] K. Fukuda and A. Prodon. Double description method revisited. In *Combinatorics and Computer Science*, volume 1120, pages 91–111. Lecture Notes in Computer Science, Springer, 1996.

[10] P. Green and A. Krieger. Recent contributions to optimal product positioning and buyer segmentation. *European Journal of Operational Research*, 41:127–141, 1989.

[11] P. Green and V. Srinivasan. Conjoint analysis in marketing research: New developments and directions. *Journal of Marketing*, 54(4):3–19, October 1990.

[12] R. Johnson. Adaptive conjoint analysis. In *Sawtooth Software Conference on Perceptual Mapping, Conjoint Analysis and Computer Interviewing*, pages 253–265. Sawtooth Software, 1987.

[13] R. Johnson. Comment on 'adaptive conjoint analysis: Some caveats and suggestions'. *Journal of Marketing Research*, 28(2):223–225, May 1991.

[14] R. Kannan, L. Lovasz, and M. Simonovitz. Random walks and an $o(n^5)$ volume algorithm for convex bodies. *Random Structures and Algorithms*, 11:1–96, 1997.

[15] R. Keeney and H. Raiffa. *Decisions with Multiple Objectives: Preferences and Value Tradeoffs*. Wiley, 1976.

[16] G. Lilien, P. Kotler, and K. Moorthy. *Marketing Models*. Prentice Hall, 1992.

[17] S. Nash and A. Sofer. *Linear and Nonlinear Programming*. McGraw-Hill, 1996.

[18] R. Pindyck and D. Rubinfeld. *Microeconomics*. Prentice Hall, 1997.

[19] T. Saaty. How to make a decision: The analytic hierarchy process. *Interfaces*, 24(6):19–43, 1994.

[20] G. Sonnevend. An "analytical centre" for polyhedrons and newlasses of global algorithms for linear (smooth, convex) programming. In *System modelling and optimization (Budapest, 1985)*, pages 866–875. Springer, Berlin, 1986.

[21] G. Tewari and P. Maes. Design and implementation of an agent-based intermediary infrastructure for electronic markets. In *ACM Conference on Electronic Commerce, EC'00*, pages 86–94, 2000.

[22] L. Valiant. The complexity of computing the permanent. *Theoretical Computer Science*, 8(2):189–201, 1979.

[23] Y. Zhang. Computational experience with the maximum-volume ellipsoid problem. In *Informs National Meeting*, October 1998.

Expt	Alg	5 Queries				10 Queries				15 Queries				20 Queries			
		Vol metric	Pos of top	Correct in top		Vol metric	Pos of top	Correct in top		Vol metric	Pos of top	Correct in top		Vol metric	Pos top	Correct in top	
				3	5			3	5			3	5			3	5
1	A1	-0.3	19	2	3	-0.7	2	3	4	-0.1	1	3	5	1.3	1	3	5
	A2	-0.4	7	2	3	-0.2	1	3	4	0.1	1	3	4	1.3	1	3	4
	A3	4.0,-1.6	72,13	1,2	2,3	4.7,0.6	24,1	1,3	2,4	7.3,5.3	4,1	1,3	4,4	11.0,7.8	2,1	2,3	4,5
2	A1	0.4	8	1	2	-0.5	1	3	4	-3.4	1	3	5	-2.1	1	3	5
	A2	0.5	4	1	3	0.1	1	3	4	0.5	1	3	5	-0.6	1	3	5
	A3	4.6,-1.9	13,1	1,3	2,4	7.2,2.7	3,1	2,3	3,4	11.2,7.3	1,1	1,3	3,4	16.1,7.2	1,1	1,3	3,5
3	A1	0	3	3	4	-0.7	1	3	5	0	2	3	5	1.7	1	3	5
	A2	0	11	2	3	0	1	3	4	1.0	1	3	5	0.8	1	3	5
	A3	3.3,-9.7	41,2	0,3	0,4	6.5,-4.7	3,1	3,3	4,4	10.0,-2.7	2,1	3,3	4,5	14.1,2.0	2,1	3,3	4,5
4	A1	0	2	2	4	0.6	2	2	5	-0.3	2	2	5	-0.8	2	3	5
	A2	0.5	3	2	3	1.5	2	3	5	2.5	1	3	5	3.2	1	3	5
	A3	2.1,0.7	8,1	1,3	2,5	6.0,4.6	3,1	2,3	4,5	8.8,-1.7	2,1	2,3	5,5	13.5,3.3	3,1	2,3	5,5

Table 3: Summary of experimental results

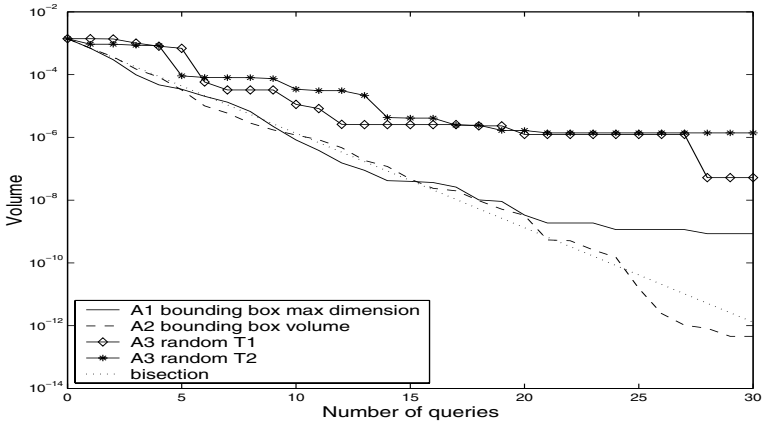

Figure 4: Volume of permissible region (Experiment 1)

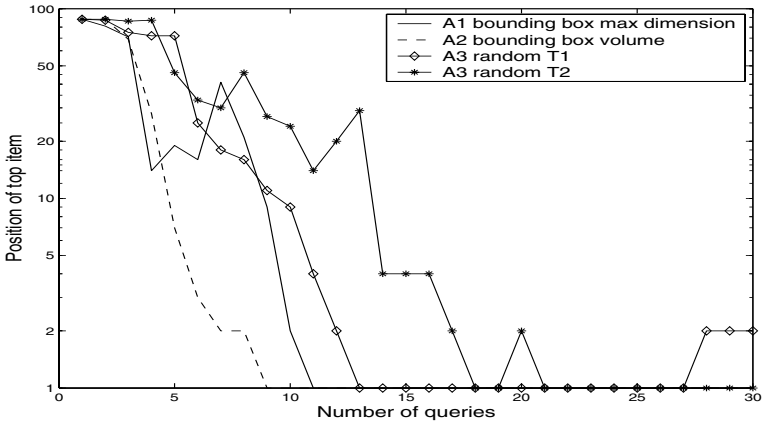

Figure 5: Position of top item (Experiment 1)

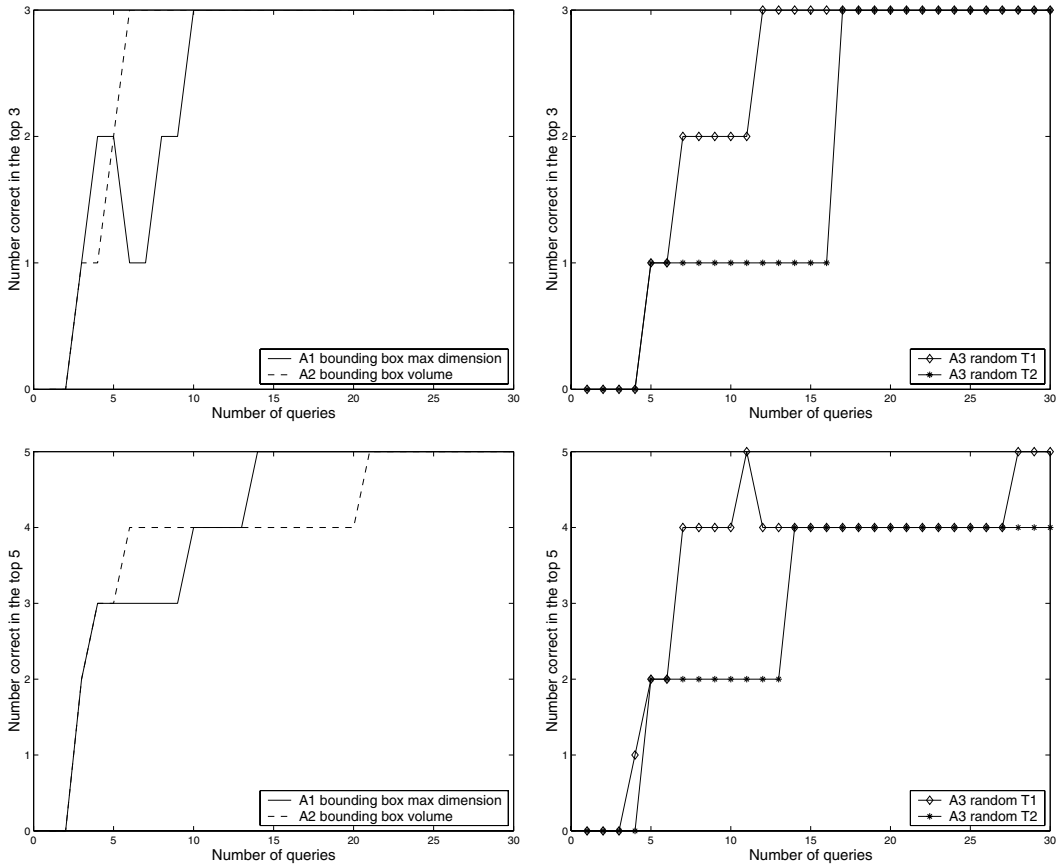

Figure 6: Comparison of algorithms using top-3 and top-5 metrics (Experiment 1)