# ADAPTIVE PREFERENCE ELICITATION FOR TOP-$K$ RECOMMENDATION TASKS USING GAI-NETWORKS

Sergio Queiroz
LIP6–Paris 6 University
8 rue du Capitaine Scott, 75015 Paris, France
email: sergio.queiroz@lip6.fr

**ABSTRACT**
The enormous number of questions needed to acquire a full preference model when the size of the outcome space is large forces us to work with partial models that approximate the user's preferences. In this way we must devise elicitation strategies that focus on the most important questions and at the same time do not need to enumerate the outcome space. In this paper we focus on adaptive elicitation of GAI-decomposable preferences for top-$k$ recommendation tasks in large combinatorial domains. We propose a method that interleaves the generation of top-$k$ solutions with a heuristic selection of questions for refining the user preference model. Empirical results for a large combinatorial problem are given.

**KEY WORDS**
knowledge representation, preference elicitation, graphical models, recommender systems, GAI networks

## 1 Introduction

Often the quantity of information to be taken into consideration in a decision making setting largely surpasses the decision maker's (DM) capacity of analyzing it. The DM may be in front of an enormous universe of alternatives, maybe even impossible to enumerate in practice. This scenario has stimulated the development of numerous applications in artificial intelligence, with the goal of automating or aiding the decision process. To take decisions, it is necessary to evaluate and compare the different available alternatives to the DM, that is, we need to know his *preferences*. In some applications, like many recommender systems, the DM exprimes his opinions about some alternatives (in a explicit or implicit manner) and the application will try to find among the available objects in a database the ones that are more probable to be liked by him. However, these approaches are only possible when the objects are available in a standardized form, as for example CDs, books, and DVDs[1]. When the objects are composed by configurable attributes, the enumeration of all possible configurations may be prohibitive. This condition characterizes the problems where the space of alternatives has a combinatorial structure, i.e. it is a Cartesian product of variables. In this case, an efficient form of representing and reasoning with preferences over combinatorial domains is needed.

In decision theory, utility functions are often used to represent the preferences of a DM. Utility functions are a powerful form of representing preferences, they furnish a quantitative measure for the utility of an object and can also capture the DM attitude towards risk and uncertainty. However, the process to obtain the information needed to build an utility function is time consuming and demands a great effort from the DM. The cognitive effort needed for the utilization of well-founded decision models has been one of the most important barriers for their adoption in some domains, as this effort may be too large relative to the value of the solution or simply time may be limited [2].

A good compromise between model simplicity and representativity is reached by GAI (*generalized additive independence*) models. Having the DM preferences modeled as GAI functions brings many benefits. The descriptive power of such models allows interactions between attributes while preserving decomposability (provided that the preferences exhibit some structure). In this way, preferences may be compactly stored. Moreover, given an utility function, we can easily compute the utility of an item, allowing us to quickly calculate its value. But still, there is no easy way to elicit GAI functions. For some problems, a qualitative model, like a TCP-net, may be used as intermediary and after compiled into a GAI function (see [3]). However, for some more elaborated decision problems utilities might significantly outperform qualitative models due to their higher descriptive power [4]. Elicitation processes directly based on GAI utilities have been proposed, first in the context of decision making under risk [5] and after for the certainty case [6]. Even though these elicitation mechanisms are general purpose, they still require a great number of questions.

In this work we use multivariate Gaussian priors over the GAI-decomposable utility functions to guide the elicitation. We treat the case under certainty, for combinatorial domains. The elicitation process is focused on quickly obtaining the top-$k$ configurations, a fundamental task for recommender systems (this departs from previous works on elicitation of GAI-decomposable preferences like [7, 8]). For this, the elicitation process (presented in Section 4) interleaves a top-$k$ algorithm for GAI-networks with heuristics based on selecting and updating the DM's preference model. Empirical results for a large combinatorial domain are presented.

## 2 Related Work

Ha et Haddawy [9] have described an incremental elicitation method for additive utility functions. They employ a heuristic that at each interaction asks the user a question that will reveal a great number of Pareto-dominated alternatives, which are then eliminated from the set of possible solutions.

Chajewska *et al.* [10] have considered the case where we cannot assume any independence between attributes (i.e. the utility function is not decomposable) and a database of utility functions is available. In this case, questions about full alternatives (i.e. configurations involving all attributes) are posed to the user in order to identify a *cluster* of users to which she can be associated. The questions to pose are defined by a decision tree that partitions the users into different clusters.

Chajewska, Koller and Parr [7] present a method to incrementally elicitate GAI functions by using prior distributions for the utility functions. The question to ask at each interaction with the DM is defined in an adaptive manner by a greedy algorithm based on the expected improvement after obtaining the answer for the next question. Braziunas and Boutilier [11] take advantage of the GAI structure to show how this can be done by asking local questions (concerning subsets of attributes).

Boutilier *et al.* [12] assume that instead of having prior distributions for the GAI utility functions, we have limits for their parameters. They present an elicitation procedure based on a mininum max-regret criterion. It tries to choose the alternative $x$ that has the minimum max-regret, where max-regret is the largest quantity we may regret by choosing $x$ (and allowing the utility function to vary within the limits of its parameters).

## 3 GAI Preferences

Suppose that we want to model the preferences of a DM w.r.t. meals composed by a main course ($M$), a wine ($W$) and a dessert ($D$). We have as options: $M$ = {beefsteak, fish}, $W$ = {red wine, white wine}, $D$ = {cake, ice cream}. Our hungry DM adopts the following directives w.r.t. meals:

1. It is essential for me to match the wine with the main course. I take beefsteak with red wine and fish with white wine.
2. I equally like cake and ice cream, but I prefer to take an ice cream for dessert when I eat beefsteak in order to avoid a too heavy meal.
3. Normally I would prefer beefsteak to fish.

These preferences induce the following preference relation for the possible alternatives: (beefsteak, red wine, ice cream) $\succ$ (beefsteak, red wine, cake) $\succ$ (fish, white wine, ice cream) $\sim$ (fish, white wine, cake) $\succ$ (beefsteak, white wine, ice cream) $\succ$ (beefsteak, white wine, cake) $\succ$ (fish, red wine, ice cream) $\sim$ (fish, red wine, cake), where $\succ$ means "is preferred to" and $\sim$ means "is indifferent to". Note that this problem has a multidimensional structure, its set of alternatives $\mathcal{X}$ is the Cartesian product of the *attributes* $M \times W \times D$.

Under some mild hypotheses [13], a preference relation can be expressed by a function $u : \mathcal{X} \mapsto \mathbb{R}$, that we call an *utility function*. With a utility function we can quickly compare alternatives, all we need is to calculate their utility values and compare them. Due to the combinatorial size of $\mathcal{X}$, it is often impossible to elicit and store a function that maps each value in $\mathcal{X}$. In practice, however, the preference relation often exhibits some independence between attributes which means that me may use a simpler, decomposable, utility function.

The most widely used model for decomposable utility functions is the additive model [14]. It assumes that the preferences over each attribute are independent of the other attribute values. In this way, the utility function over a combinatorial set $\mathcal{X} = X_1, ..., X_n$ may be decomposed as a sum of single-attribute *utility functions*, that is:

$$u(x) = \sum_{i=1}^{n} u_i(x_i).$$

Unfortunately this is a too strong of an assumption. In our example, it means that the utility function for our DM should be decomposed as $u(m, w, d) = u_1(m) + u_2(w) + u_3(d)$. However, we clearly see that this is not the case; the DM's preferences over wine and dessert depend on the main dish.

The *generalized additive models* (GAI) allow the representation of preferences where individual attributes are not additively independents but (possibly nondisjoint) subsets of them are [15, 16].

**Definition 1** *Let $\mathcal{X} = X_1 \times X_2 \times ... \times X_n$; $Z_1, ..., Z_k$ be subsets of $N = \{1, ...n\}$ such that $N = \bigcup_{i=1}^{k} Z_i$ and $X_{Z_i} = \{X_j : j \in Z_i\}$. A GAI utility function $u$ over $\mathcal{X}$ may be decomposed as:*

$$u(x_1, ..., x_n) = \sum_{i=1}^{k} u_i(x_{Z_i})$$

*where each $u_i$ is a function $u_i : X_{Z_i} \mapsto \mathbb{R}$*

The preferences of our DM may be represented by the GAI decomposition: $u(m, w, d) = u_1(m, w) + u_2(m, d)$. Figure 1 shows preference tables for $u(\cdot)$ that reflect the DM's preferences in the example.

The GAI-networks, a graphical model that directly represents GAI decompositions, were introduced in [5]. They are similar to junction trees used in bayesian networks.

**Definition 2** *Let $u(x_1, ..., x_n) = \sum_{i=1}^{k} u_i(x_{Z_i})$ be a GAI utility function over $\mathcal{X}$. A GAI-network that represents $u$ is a undirected graph $G = (V, E)$ such that:*

| $u_1$ | beef | fish |
|---|---|---|
| white | 7 | 8 |
| red | 10 | 4 |

| $u_2$ | beef | fish |
|---|---|---|
| cake | 9 | 10 |
| ice cream | 10 | 10 |

Figure 1. Preference tables for the meal example.

1. $V = \{X_{Z_1}, ..., X_{Z_k}\}$. *The vertices* $X_{Z_i}$ *are called* cliques. *For each vertex* $X_{Z_i}$ *we associate the term* $u_i$ *from the utility function* $u$*;*
2. $\forall \langle X_{Z_i}, X_{Z_j} \rangle \in E \Rightarrow Z_i \cap Z_j \neq \emptyset$. *The edges* $\langle X_{Z_i}, X_{Z_j} \rangle$ *are labeled by* $X_{T_{ij}}$*, where* $T_{ij} = Z_i \cap Z_j$. $X_{T_{ij}}$ *is a* separator*;*
3. $\forall X_{Z_i}, X_{Z_j}$ *such that* $T_{ij} \neq \emptyset$*, there is a path between* $X_{Z_i}, X_{Z_j}$ *such that for every vertex* $X_{Z_k}$ *in this path* $T_{ij} \subseteq Z_k$ *(running intersection property).*

The similarity of GAI-networks to bayesian networks is very useful, as they allow us to bring into the utility realm some of the techniques successfully used for reasoning and eliciting probabilities, so that we could also efficiently elicitate and reason with utilities.

# 4 Adaptive Elicitation of GAI-functions to Find Top-*k* Configurations

One of the greatest obstacles to the adoption of preference representation models based on utility functions is the amount of information needed to build them. The elicitation process often requires an enormous number of questions and still, such questions may be difficult to answer.

In order to avoid having to elicitate the DM's utility function completely before being able to reason with it, we need methods that are able to reason with a partial preference specification. It is reasonable to assume that some regions of the utility function are more important than others for the determination of the best alternatives for the DM. In this way, we can think of incrementally eliciting this function, giving priority to its more important regions. This poses two problems: *how to choose the points to elicitate* and *how to reason with an incomplete function*. As we don't know the function, we cannot know with certainty which are its more important points. Therefore, we need to resort to heuristic methods.

The heuristic to use depends on the problem in question. We need to take into consideration the nature of the desired solution (do we look for the best solution, a list of best solutions?). Often, we expect that a decision aid system will not only be able to find the best solution for an user, but also that it can show what are the $k$-best alternatives for the user. For instance, for a recommender system, Herlocker [17] identifies two essential features that such a system should have: (1) the system should be able to list the $k$-best recommendations, (2) the system should be capable of giving an absolute utility value for any arbitrarily chosen item.

Contrary to previous works on elicitation of GAI models, we assume that the goal of the system is to discover the $k$-best solutions for the DM. In this way, we propose methods that focus on the improvement of the $k$-best alternatives.

The considered scenario is an interactive process where the system queries the user about her utility function, thus refining its knowledge about her. At each step, the system is capable of presenting the user the top-$k$ recommendations for the current preference model.

We use variable elimination to determine the best $k$ alternatives. Normally, the algorithm of variable elimination is used to determine only the best alternative. However, Nilsson [18] has proposed a method to enumerate the $M$ most probable configurations in a bayesian network. We have adapted this algorithm to work with GAI-networks.

We assume that the system has a database of user preferences. These preferences are GAI-decomposable, and the users can be placed into different clusters (classes of users). We should be able to use this information to focus the elicitation process of a new user on questions that will allow us to quickly identify her preferences. Notice that clusters can differ both in structure and values, as observed by [19]. However, here we assume that all clusters share the same decomposition structure, differing only in values. If this was not the case, the user could be first asked questions in order to determine which structure between the candidates were the most appropriate for her, in a two phase elicitation process: structure selection, followed by utility elicitation.

## 4.1 Elicitation Queries

At each interaction with the user, we ask her a *bound query* in which she tells if his utility for a configuration lies above a certain level $l$. A bound query is equivalent to a *standard gamble query (SGQ)* where it is asked if the user prefers configuration x to a gamble in which the best outcome $\mathrm{x}^\top$ happens with probability $l$ and the worst outcome $\mathrm{x}^\perp$ happens with probability $1 - l$. As described in [11], the elicitation process in GAI models can be separated into local elicitation, where the configurations proposed to the user involve local utility factors, and global scaling where calibration is attained by using full best and worst outcomes.

The goal of an elicitation strategy is to attain the top-$k$ configurations for the user (ordered by decreasing utility) using as few questions as possible. In our setting, there are no probabilities associated to each outcome, that could be used to focus the questions on outcomes with highest probabilities. Therefore we focus on outcomes that are highly placed in the list of configurations presented to the user, as these have a higher probability of being noticed by her.

The following algorithm summarizes the elicitation process:

**Function** `Elicit_Preferences`($\mathcal{M}$)
01 In the beginning, set GAI-net to the average utility given the models $\mathcal{M}$
02 cycle through:
03 compute `k-best` solutions according to current GAI-net
04 query the user using current strategy (and potentially the k-best solutions found)
05 update Elicited_bounds according to user's answer
06 call `Update_GAI`

### 4.2 The Max Variance Strategy

The first strategy considered is the one that focus on reducing the model variance. This suggests a straightforward strategy where at each interaction the user is asked if the utility for the local configuration with largest variance is larger than its expected value (i.e. its mean).

### 4.3 The Best Solution Strategies

The max variance strategy tries to quickly reduce the model variance as a whole, not considering the fact that the most important queries to ask are those about configurations that have a larger chance of being highly ranked by the user. Therefore, an alternative approach is to focus on refining the knowledge about the best ranked solutions.

The *best solution highest utility strategy (BSHU)* asks the local utility factor configuration (for the solution currently believed to be the best one) that most contributes to the total utility. That is, given the current best solution $x^{(1)}$, we ask the question corresponding to the local utility $u_i(x^{(1)}_{Z_i})$ such that $u_i(x^{(1)}_{Z_i}) \geq u_j(x^{(1)}_{Z_j})$ for all $j$. If all local utility configurations for the current best solution have already been queried, we use the $2^{nd}$ best solution $x^{(2)}$ and so on.

Instead of asking the local configuration that most contributes to the total utility of the current best solution, the *best solution maximum variance strategy (BSMV)* asks the local configuration with the highest variance for the solution.

### 4.4 Updating the target GAI

After getting an answer from the user, the utility model should be updated. The following algorithm is used for this task:

**Function** `Update_GAI`(GAI-net, $\mathcal{M}$, Elicited_bounds)
01 for each candidate model $\mathcal{M}_j$ in $\mathcal{M}$ do
02 calculate probability of observing Elicited_bounds
03 done
04 let $\mathcal{M}_*$ be the model with highest probability
05 for each local utility configuration $u_i(x_{Z_i})$ in GAI-net
06 if $u_i(x_{Z_i})$ has Elicited_bounds then
07 set $u_i(x_{Z_i})$ to the expected value of $\mathcal{M}_*$ given Elicited_bounds
08 else
09 set $u_i(x_{Z_i})$ to the expected value of $\mathcal{M}_*$
10 endif
11 done

When we have independent multivariate gaussians as models for each subutility function, the probability of observing the elicited bounds given a model $\mathcal{M}_j$ can be quickly calculated (line 02). Given that we have for a elicited configuration the highest bound $l_{up}$ and the lowest bound $l_{down}$, the probability of observing these values given $\mathcal{M}_j$ is $F_j(l_{up}) - F_j(l_{down})$ where $F(\cdot)_j$ is the cumulative distribution for $M_j$ (at the elicited configuration).

We estimate the expected value $\mu$ for the configurations with elicited bounds (line 07) by sampling from the truncated distribution (i.e., the selected model distribution given the elicited bounds). In order to bound the number of samples, we establish a confidence interval. The $100(1-\alpha)$ percent confidence interval for $\mu$ after $n$ samples is given by $\hat{\mu}(n) \pm t_{n-1,1-\alpha/2}\sqrt{\frac{\hat{\sigma}^2(n)}{n}}$, where $t_{n-1,1-\alpha/2}$ is the upper $1-\alpha/2$ critical point for the $t$ distribution with $n-1$ degrees of freedom. In the experiments, we used a confidence interval of length 0.01 with 95% of confidence.

## 5 Empirical Analysis

We tested the elicitation strategies on synthetically generated large problems with varying number of variables and GAI structure. As the obtained results were similar, we show then for a test problem with a domain of 30 variables, each one with 5 possible values (i.e. $5^{30}$ possible configurations). This test problem has 23 subutility functions, being 1 with 5 variables, 2 with 4 variables, 14 with 3 variables and 6 with 1 variable. Each subutility function has utilities values in the interval $[0, 1]$.

### 5.1 Evaluation Criteria

To evaluate the quality of the ranked list returned in the experiments, we use an approach inspired by the work of [20] when evaluating collaborative filtering algorithms. The expected utility of a ranked list for an user $a$ is the probability that user $a$ will see the presented items times their utility. In order to estimate how likely it is that the user will see an item on a ranked list it is assumed that each successive item in the list is less likely to be viewed by the user with an exponential decay. Then the expected utility of a ranked list of items for user $a$ (sorted by index $j$ in order of decreasing $u^{a,j}$) is given by:

$$R^a = \sum_j \frac{u^{a,j}}{2^{(j-1)/(\alpha-1)}} \quad (1)$$

where $\alpha$ is the halflife–the number of items on the list such that the user has 50% chance of viewing that item. When scoring a list of items returned for an user, we apply the equation 1 using the user's real $u^a$. The final score is: $S^a = R^a/R^a_{max}$, where $R^a_{max}$ is the maximum achievable $R^a$ that is, the value for the top items ranked in the correct order. We refer to $(1 - S^a)$ as the *Breese error*. Given the transformation it is always in the interval $[0, 1]$. In our experiments we consider the *average Breese error* for the set of users. We used a half-life of 5 and measured the *Breese error* over the 50 best ranked items.

### 5.2 User class separation

When the classes of users are well separated, identifying to which model the user most probably belongs is an easy task. In these cases, after few questions the model chosen (line 4 of `Update_GAI`) will no longer change.

To evaluate the elicitation performance for more challenging data, we have to consider cases where the classes are interposed. In order to simulate this, we introduce dependencies among the models. Instead of randomly choosing the values for the mean of each multivariate gaussian in the model, we set for each model $\mathcal{M}_j$, $j \neq 1$, $\mu_{\mathcal{M}_j} = \mu_{\mathcal{M}_1} + \text{Normal}(0, \eta)$, where $\text{Normal}(0, \eta)$ is a sample from a multivariate normal with mean 0 and covariance matrix $\eta I$. As $\eta$ goes toward 0, the cluster separation diminishes.

### 5.3 Empirical Results

We generated 3 different clusters, each one with 30 individuals. To create a cluster of users, for each subutility function we first generate a vector of values randomly distributed in the interval $[0, 1]$. This vector will be the mean of the multivariate gaussian for the model. Then we generate a diagonal covariance matrix with values randomly chosen in an interval (we used the interval $[0.05, 0.2]$. The subutility functions for users in the cluster will be samples from this multivariate gaussian (truncated in the interval $[0, 1]$).

We evaluated the effectiveness of our query strategies by measuring the average Breese error over the 90 users. Figure 2 shows the observed average Breese error over 200 questions using the algorithm `Elicit_Preferences`. In Figure 2(a) the clusters are completely separated, i.e., the means for the models were independently generated. Figures 2(b) and 2(c) show the results for correlated models. Figure 2(b) depicts high correlated models, with $\eta = 0.05$ whereas Figure 2(c) shows the observed results for low correlated models, with $\eta = 1.25$.

We see that the strategies that focus on the best $k$ solutions perform much better than undirected strategies. In all cases the BSMV strategy performed the fastest reduction of the average Breese error, followed by the BSHU strategy. The *max variance* strategy performed particularly badly, being even inferior to a *random strategy* (where the next question is randomly chosen). This shows the importance of focusing in regions of the utility space that will most probably contribute to the k best solutions instead of trying to improve the model as a whole.

Figure 2(b) shows that for highly correlated models the Breese error decreased in a much slower rate than in more separated models. This occurred because in these cases the algorithm in `Update_GAI` (line 4) frequently switches back and forth between the different candidate models. Therefore, many questions were made for suboptimal points in the utility space. To deal with this problem, we introduced an *inertia* term by modifying line 4 of `Update_GAI` to only switch to a new model $\mathcal{M}_*$ if it was the best one for the last $i$ interactions. Figure 3 shows the results for the same dataset using $i = 3$. We can see that the Breese error decreased with a higher rate for the high correlated models and the Best Solution strategies, whereas the difference in performance for low correlated models was not significant.

## 6 Conclusion

We have shown that query strategies focused the top-$k$ configurations of a GAI preference model have a very good performance, rapidly reducing the Breese error after a small number of questions (given the size of the problem). All computations were made in real-time, with instant perceived response (less than 1 second) for our test problem. This work can be extended in a number of directions, we would like to include new types of question, like comparison of outcomes, or allowing the user to reorder the top-$k$ suggestions as feedback. Also, the complexity of answering a question could be taken into consideration. For example, priority could be given to the questions involving a smaller number of attributes.

## References


[1] F. Ricci, A. Venturini, D. Cavada, N. Mirzadeh, D. Blaas, and M. Nones, "Product recommendation with interactive query management and twofold similarity," in *Proc. 5th International Conference on Case-Based Reasoning, ICCBR 2003*, ser. LNCS, vol. 2689. Springer, 2003, pp. 479–493.

[2] V. A. Ha and P. Haddawy, "Toward case-based preference elicitation: Similarity measures on preference structures." in *"Proc. 14th Conference on Uncertainty in Artificial Intelligence (UAI)"*, 1998, pp. 193–201.

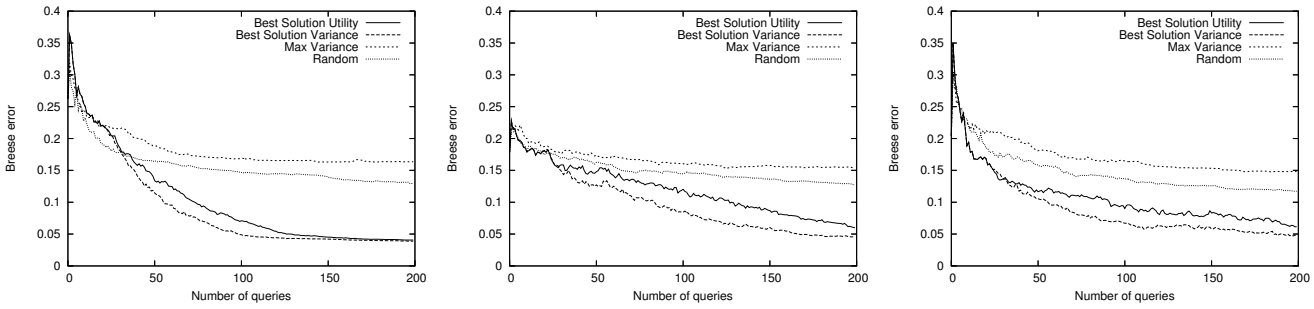
[3] R. I. Brafman, C. Domshlak, and T. Kogan, "Compact value-function representation for qualitative preferences," in *UAI-2004*, 2004, pp. 51–58.

[4] C. Boutilier, F. Bacchus, and R. I. Brafman, "Ucp-networks: A directed graphical representation of conditional utilities." in *UAI*, 2001, pp. 56–64.

[5] C. Gonzales and P. Perny, "GAI networks for utility elicitation," in *KR'04*, 2004.

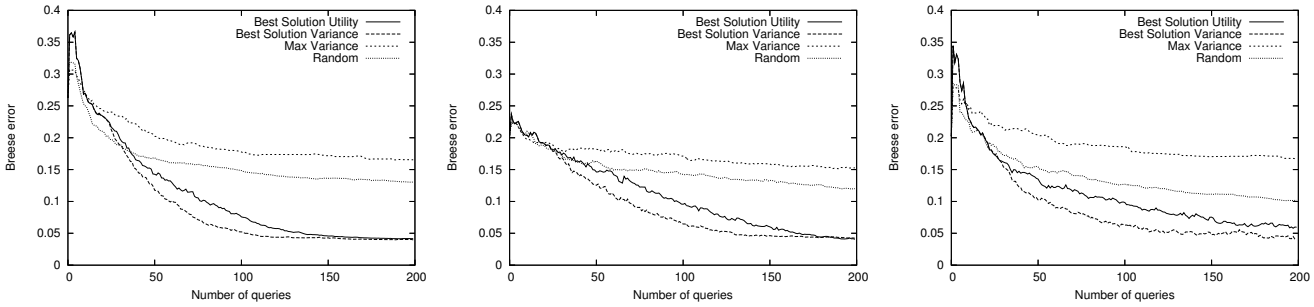
[6] ——, "GAI networks for decision making under certainty," in *IJCAI'05 – Workshop on Advances in Preference Handling*, 2005.

[7] U. Chajewska, D. Koller, and R. Parr, "Making rational decisions using adaptive utility elicitation." in *AAAI/IAAI*, 2000, pp. 363–369.




(a) Complete separation (b) $\eta = 0.05$ (c) $\eta = 1.25$

Figure 2. Average Breese error over 90 users from 3 clusters (30 users by cluster), no inertia




(a) Complete separation (b) $\eta = 0.05$ (c) $\eta = 1.25$

Figure 3. Average Breese error over 90 users from 3 clusters (30 users by cluster), inertia 3


[8] C. Boutilier, R. Patrascu, P. Poupart, and D. Schuurmans, "Regret-based utility elicitation in constraint-based decision problems." in *IJCAI*, 2005, pp. 929–934.

[9] V. A. Ha and P. Haddawy, "Problem-focused incremental elicitation of multi-attribute utility models." in *UAI*, 1997, pp. 215–222.

[10] U. Chajewska, L. Getoor, J. Norman, and Y. Shahar, "Utility elicitation as a classification problem," in *UAI*, 1998, pp. 79–88.

[11] D. Braziunas and C. Boutilier, "Local utility elicitation in gai models," in *UAI-05*. Arlington, Virginia: AUAI Press, 2005, p. 42.

[12] C. Boutilier, R. Patrascu, P. Poupart, and D. Schuurmans, "Regret-based utility elicitation in constraint-based decision problems," in *IJCAI*, 2005, pp. 929–934.

[13] G. Debreu, "Continuity properties of paretian utility," *International Economic Review*, vol. 5, pp. 285–293, 1964.

[14] R. L. Keeney and H. Raiffa, *Decisions with multiple objectives: preferences and value tradeoffs*. John Wiley and Sons, Inc., 1976.

[15] P. C. Fishburn, "Interdependence and additivity in multivariate, unidimensional expected utility theory," *Int. Economic Review*, vol. 8, pp. 335–342, 1967.

[16] F. Bacchus and A. J. Grove, "Graphical models for preference and utility." in *UAI*, 1995, pp. 3–10.

[17] J. L. Herlocker, "Understanding and improving automated collaborative filtering systems," Ph.D. Thesis, CS Dept., University of Minnesota, USA, 2000.

[18] D. Nilsson, "An efficient algorithm for finding the M most probable configurations in probabilistic expert systems," *Statistics and Computing*, vol. 8, no. 2, pp. 159–173, 1998.

[19] U. Chajewska and D. Koller, "Utilities as random variables: Density estimation and structure discovery." in *UAI*, 2000, pp. 63–71.

[20] J. S. Breese, D. Heckerman, and C. M. Kadie, "Empirical analysis of predictive algorithms for collaborative filtering." in *UAI*, 1998, pp. 43–52.