RC22774 (W0304-103) April 11, 2003
Computer Science

IBM Research Report

Polyhedral Sampling for Multiattribute Preference Elicitation

S. Ghosh
Cornell University
Ithaca, NY 14853

Jayant R. Kalagnanam
IBM Research Division
Thomas J. Watson Research Center
P.O. Box 218
Yorktown Heights, NY 10598

Research Division
Almaden - Austin - Beijing - Delhi - Haifa - India - T. J. Watson - Tokyo - Zurich

Polyhedral Sampling for Multiattribute Preference Elicitation

Soumyadip Ghosh*and Jayant Kalagnanam †

Abstract

Multiattribute auctions are now routinely used for B2B negotiations in electronic procurement. A basic requirement for running multiattribute auctions is the utility function of the buyer that trades off nonprice attributes against price. Such a utility function can then be used to design a (strategic) scoring function that communicates to the suppliers how the buyer will evaluate multiattribute bids. One approach to eliciting the preference strucure is to model it as a polyhedra in the space of the parameters and design pairwise comparisons to quickly narrow down the feasible region. In order to perform this in real time, we need efficient techniques for estimating the centroid and a cut that is perpendicular to the "longest axis" of the polyhedra thereby minimizing the feasible region for the parameters. In this paper, we present the use of a "hit-and-run" algorithm for sampling uniformly from a polyhedra. We tailor the use of this algorithm to also produce a cut that approximately minimizes the volume of feasible polyhedra. The advantage of this technique is its relative simplicity - it relies only on matrix algebra and avoids the use of nonlinear optimization techniques. Computational results suggest that this method is fast and accurate.

1 Introduction

Electronic reverse auctions are now being increasingly used in electronic procurement. Multiattribute reverse auctions are common in B2B negotiations and are used in two

*ORIE, Cornell University, NY; sdghosh@orie.cornell.edu
†IBM T. J. Watson Center, NY; jayant@watson.ibm.com

flavors:

1. Iterative multiattribute auctions where a scoring function for evaluating multi-attribute bids is published at the beginning of the auction. This requires that the *utility function* of the buyer is elicited before starting the auction and an appropriate scoring function derived for communicating with the suppliers.

2. Sealed bid auctions where a scoring function may or may not be communicated. If a scoring function is not communicated as part of the auction initialization, then utility elicitation can be done after the bids are received to rank order them.

The preference elicitation exercise that is used in these contexts is to conduct an interview where the queries are pairwise comparisons between two virtual or real bids. Based on the responses (ordinal or cardinal) the weights for different attributes is estimated. Typically, when the interview is done after bids have been submitted, the queries are restricted to actual bids.

A common approach used to model the utility function[1] is to use an additive form with different weights associated with each attribute. Such a function is characterized by a polyhedra in the space of weight parameters. A key aspect of building tools for preference elicitation is to provide a method for designing queries that correspond to "cuts" through the polyhedra so as to minimize the volume of the feasible region for the weight parameters.

Previous approaches have used nonlinear programming techniques to design these queries. In online situations with high dimensions this could prove cumbersome. We propose the use of sampling techniques to approximate these computations. The novelty in this paper is the introduction of the *hit-and-run alogorithm* [7] to sample a polyhedra uniformly (using ideas from Markovian sampling techniques) and derive as a by product the hyperplane that minimizes the feasible region. We provide experimental results for both the types of responses discussed above. The approach provides excellent convergence properties in terms of identifying the top ranked bid from a given set of bids.

[1] based on multi-attribute utility theory

The paper is organized as follows. Section 2 provides an overview of the interview method used for multiattribute preference elicitation and outlines the two core problems of the method. Section 3 introduces the hit-and-run algorithm and tailors it's use to solve the core problems introduced in Section 2. Section 4 provides computational results that illustrate the efficacy of this method. Section 5 provides conclusions.

2 The Method

The interview process used for preference elicitation consists of presenting the buyer with a pair of bids (B_1, B_2) and asking for a response. Let $x = (y_1, \ldots, y_n)$ represent a bid with n attributes, each fixed at level y_i. Suppose $x_i = f_i(y_i)$ represents the worth of the i^{th} part to the customer when fixed at level y_i. We now assume that the individual functions f_i are known , and the overall utility of the customer over all attributes is additive, i.e., the total utility of a bid for the customer is $\sum_i w_i x_i$, where w_i is the relative weight associated with the i^{th} attribute. Our aim is to elicit the values of w_i.

Assuming that the utility function of the buyer is an additive function $U(\hat{x}) = \sum_i w_i f_i(y_i)$. A common assumption is that the functions $f_i(x_i)$ is known apriori. The response can be of two types:

Type 1 Response The buyer is able to indicate exactly by how much he prefers B_1 to B_2. For example, $U(B_1) - U(B_2) = u_{12}$. This can be written as a linear equality $\sum_i w_i(x_{1i} - x_{2i}) = u_{12}$, where $x_{1i}(x_{2i})$ is the worth of the ith attribute of Bid 1 (2).

Type 2 Response Alternately, the buyer may only specify preference ordinally, i.e. $U(B_1) \geq U(B_2)$. This can be written as a linear inequality $\sum_i w_i(x_{1i} - x_{2i}) \geq 0$.

Based on a series of such responses the weights w_i are computed. At anytime in the interview, the feasible weights that are consistent with the responses to the query correspond to a convex polyhedra in the weight space. Typically the estimate of weights is computed as the centroid of this feasible polyhedra and the next query (i.e. a pair of candidate bids to be presented to the buyer for comparison) is designed so as to reduce the feasible region maximally. There are two central computations:

1. Computing the centroid of a polyhedra, and

2. Computing a hyperplane that minimizes the feasible region for the weights maximally.

Consider the parameter space for w_i. The feasible values for w_i is represented by a set of linear equality (or inequality) constraints which constitute a polyhedra. Each query adds another constraint thereby tightening the feasible polyhedra. Suppose that the w_i have a set of prior constraints that they need to satisfy, and that these constraints are collectively represented in the matrix A and and the vector b, where the constraint is of the form $\sum_i A_{ji} w_i \leq, =$ or $\geq b_j$. Some obvious necessary constraints that we will impose are that $0 \leq w_i \leq 1, \ \forall i$ and that $\sum_i w_i = 1$.

The procedure is to then pose questions and based on the response from the customer impose an additional constraint on the w_i, consisting of, say, $A_{m+1,\cdot}$ and b_{m+1}. We do not at this stage impose any condition on whether the constraint should be an inequality or an equality, and in fact our method works with both cases. We continue is this fashion till the set of feasible w is narrowed down to one with sufficiently small volume.

3 Computations using the Hit-and-Run Sampling Algorithm

In this section we introduce the hit-and-run algorithm and discuss how this can be used for computing the centroid of a convex polyhedra and for generating the hyperplane that minimizes the volume fo the feasible region.

3.1 The Hit-and-Run Algorithm

The Hit-and-Run Algorithm was proposed in 1979 by [1] and independently in 1980 by [7] to sample uniformly from a given bounded convex region. This sampling method is a particular case of the generic class of Markovian sampling techniques that employ a random walk defined on the set S with a stationary distribution that is uniform

over the set S, as their basis. This particular choice of stationary distribution then ensures that if one follows the chain for a sufficiently large number of steps, then the points obtained will be approximately uniformly spread out in S.

The generic algorithm for generating feasible points in S using these methods can be described as follows:

- Choose a starting point $x_0 \in S$ and set $i = 0$.
- Generate a random direction d chosen from a distribution D over the set of all possible directions. Find the *line set* $L = S \cap \{x |\ x = x_i + \lambda d, \lambda \text{ is a real scalar } \}$ and generate a random point x_{i+1} uniformly distributed over L.
- If $i = N$, stop. Otherwise, set $i = i + 1$ and return to 2.

The line set L is just the segment of the line through x_i along the direction d that is contained in the set S. The hit-and-run algorithm in its general form uses a distribution D that has a non-zero density over the set of all unit-norm vectors, i.e., all points on the unit hypersphere. The distribution D is commonly assumed to be the uniform distribution on the hypersphere. [7] shows that if the directions d are each chosen independently from this distribution (i.e., uniformly from the unit hypersphere), then the resulting markov chain is ergodic and reversible, and also that the stationary distribution of the markov chain is uniform on the convex region S (the *stationary distribution* here is the distribution of the points x_i over S as $i \to \infty$.)

3.2 Centroid Calculation

Some methods for computing the centroid of a polyhedra is to approximate it using an *analytic center*[2]. Computing the analytic center requires solving a nonlinear optimization problem [2] [9], [8]. Another option is to approximate the centroid by a *vertex barycenter*[3] [4]. The vertex enumeration problem is $\neq$P-complete [3] and a vaible option for low dimensionality (less than 10).

[2]The analytic is a point in S that maximizes the geometric mean of the distances from the point to the faces of S

[3]Computed as the simple average of the coordinates of vertices of S

In terms of our problem, the hit-and-run algorithm gives us the ability to sample points x_i, $i = 1, \ldots, N$ from S that are approximately uniformly spread out over S. This immediately gives us a straightforward estimate $\bar{x}_i$ of the centroid of the polyhedron S as the average of the points $x_1, \ldots, x_i$ generated so far.

Note at this stage that the centroid estimation and the constraint-question generation steps do not themselves depend on which sampling method is used to sample uniformly from S as long as the samples are spread uniformly on the polyhedron. However, the practical merit of this algorithm is of course critically dependent on how fast the sampling scheme can generate points that are sufficiently uniformly spread out in S. Fortunately, the hit-and-run sampling algorithm has proven bounds on the speed of convergence to the distirbution of the sample points to the stationary uniform distribution, and in practice have been found to be faster in convergence than many of sampling schemes. [7] has shown that the difference between the probability distirbution of the m^{th} point x_m over S in the chain and the uniform distribution falls exponentially w.r.t. $m - 1$. The base of the exponent is however dependent on S and on the choice of the direction distribution D, and can be made really large for appropriately chosen S and D. The base term can however be optimized to be minimal under the right choice of D, and in fact [5] present such a result for a certain class of S.

[6] has been able to improve on the bound provided by [7] to show that the so-called *mixing time*, the time it takes the points in the chain to be approximately uniformly distributed over S, is, under the original hit-and-run choice of the direction distn D above and some appropriate preprocseeing step, is of order $O^*(n^3)$, where n is the dimensionality of the set S. Thus the hit-and-run sampling algorithm gives us an approximately uniformly spread out set of points for every order n^3 number of points sampled from the markov chain, a limit that is excellent new practically.

As mentioned above, the convergence can be further accelerated by an appropriate choice of D. [5] show that the exponential upper bound of [7] can be minimized by a direction distribution D^* namely

$$D^* =^d \frac{(Y - s)}{||Y - s||}, \tag{1}$$

where $=^d$ signifies equailty in distribution, Y is distributed uniformly on S, and s, the center of S, is defined to be the point that is equidistant from the two endpoints

of a line segment drawn through in almost any direction. The class of polyhedra for which such a center exists is quite small, but the result itself motivates the use of an approximate equivalent of the distribution in equation (1) to speed up the convergence, where the estimated centroid is substituted for s and Y is sampled independently uniformly from the prior samples generated by the markov chain. [5] call this method the Artifical Centering Hit-and-Run (ACHR) method. They provide simulational evidence to show that for many reasonable S, a notable increase in the convergence rate from the earlier version of hit-and-run can be observed. We will use this ACHR version of the hit-and-run method for our purposes.

3.3 Cut Generation - Type 1 Response

As noted before, Type 1 response leads to an equality cut. The objective is to be able to derive a method that chooses questions carefully in order to obtain the maximum reduction in the size of the region S using these additional constraints. In this sense, an equality constraint is probably more advantageous since every extra equality constraint reduces the effective dimension (the degrees of freedom) by one.

Toubia[8] address this question in their method by first approximating the bounded polyhedron S with a circumscribing ellipsoid, and then by structuring their questions such that the corresponding constraints are perpendicular to the major axis (the longest axis) of the circumscribing ellipsoid. The intuitive idea behind this is that since their questions are all equalities (type 1), the resulting feasible polyhedron after placing the extra constraint is just the intersection of the polyhedron S prior to the constraint insertion and the hyperplane that represents the constraint. Hence, one would like the hyperplanes to have the minimal possible cross-sectional "area", and the hyperplanes perpendicular to the major axis are a good approximation of such hyperplanes. However in order to compute the longest axis of the circumscribing ellipsoid Toubia et al [8] solve a nonlinear optimization problem.

We will use a similar question generation procedure in our method, but with two variations:

1. Instead of the longest axis of the circumscribing ellipsoid we identify the longest line segment that can be fitted into the convex polyhedra and choose a hyper-

plane that is orthogonal. This is ofcourse an approximation to identifying the hyperplane with the minimal cross-sectional area. Note however that so is the longest axis of the circumscribing ellipsoid. Once we identify the direction $\hat{d}$ of the longest line segment of the present S from the uniform sample points generated, and then structure a question that results in a constraint of the form $\sum_i w_i \hat{d}_i \ \leq, =, \geq \ c$, where c is a constant.

2. A natural approximation of $\hat{d}$ can be found by searching the directions $\hat{x}_i - x_i$ at every stage i of the hit-and-run point generation process. Since all directions are sampled uniformly this provides a good approximation of the longest axis provided sufficient samples are generated Figure 1 illustrates our procedure for the case where S is a two-dimensional polyhedron as shown. The best estimate of the centroid and the longest axis are also shown.

The constraint (and hence question) choice is justified by arguments similar to those described above. The fundamental differences in our approach is that the choice of the longest direction and the computation of the direction of the longest segment is identified as a by product of the hit-and-run algorithm.

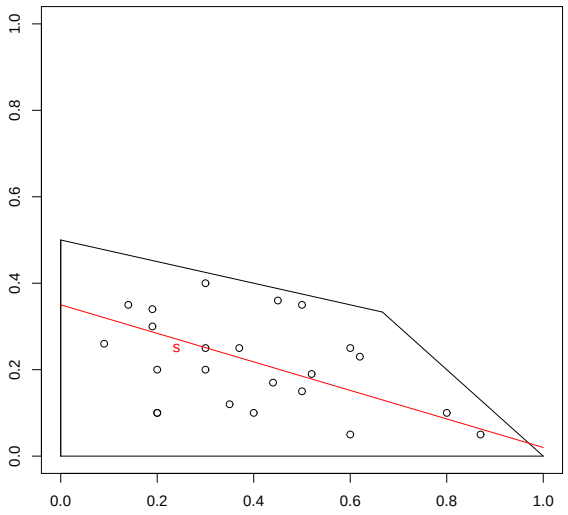

Figure 1: Illustration of Hit-and-Run for generating longest segment direction

We now recap the steps our method goes through in generating a question, given that the present feasible polytope is S, with dimensionality (degrees of freedom) n.

1. Choose an arbitrary starting point $x^0 \in S$. Let W represent the number of warmup samples ($W \geq n$). Set $m = 0$, $\hat{s} = x^0$, $M = 0$ and $d^M = 0$.

2. If $m < W$ (i.e., warmup phase), generate a random direction d^m according to the uniform distribution on the unit hypersphere. Otherwise (i.e., in the main phase), select a number a from the uniform distribution on $\{0, 1, \ldots, m-1\}$ and set $d^m = (x^a - \hat{s})/||x^a - \hat{s}||$.

3. Find the *line set* $L = S \cap \{x|\ x = x^m + \lambda d^m, \lambda \text{ is a real scalar }\}$ and generate x^{m+1} uniformly distributed over L.

4. Find the line set $L' = S \cap \{x|\ x = \hat{s} + \lambda d^m, \lambda \text{ is a real scalar }\}$. Determine $length(L')$, the length of the line set, which is the range of the λ values in the L' definition. If $length(L') > M$, then set $d^M = d^m$.

5. Set $m = m + 1$ and $\hat{s} = (m\hat{s} + x^{m+1})/(m+1)$.

6. If m is sufficiently large, return $\hat{s}$ and d^M as the estimates for the centroid and longest axis. Else, Go to Step 2.

The algorithm first goes through a warmup phase, where the aim is to just generate sufficient points in order to get a reliable estimate of the centroid $\hat{s}$ for use in the main phase. In both the phases, we also keep track of the maximum chord-length (length of the line segments L').

Note that at any query-stage, if the queries are returned with an equality, then this automatically decreases the degrees of freedom of our problem by one. In terms of the polyhedron, this means that one of the weight-variables can be eliminated using the equality. This also means that if the replies are consistent, then one needs only a maximum of $n - 1$ such queries to converge to the right weights (remember that we also need the condition $\sum w_i = 1$). In the case where the replies are ordinal, or if the respondent is able to specify only upper and lower bounds on the difference in utilities of the bids being compared, then this is no longer the case. We investigate all three cases below in our simulation tests.

3.4 Cut Generation - Type 2 Response

Type 2 responses lead to inequality constraints and have been used in the context where the bids are known apriori [4]. In this setting the cut generation is restricted to bid comparisons between the known bids. For example, if a buyer in a procurement auction sent out a RFQ and receives 10 bids in response, a total of 45 bid comparisons (and hence directions) are considered. The cut generation problem is now reduced to choosing (among the 45 comparisons) the bid pair whose comparison would bisect the volume of the polyhedra, i.e. the volume of the polyhedra on either side of the cut should be equal. One approximate technique for computing the volumes (used in [4]) is use a bounding hypercube.

We propose an alternate approach for this volume approximation by using the sample generated from the hit-and-run algorithm. The difference in the volume of the two regions generated by any cut can be approximated by the difference in the number of samples in each region. This technique for volume computation can be embedded into the step cut generation as follows:

1. Given a bid set $\mathcal{B}$, compute all allowed cuts $\mathcal{C}$ using pairwise comparisons.
2. Generate a sample of points $x_i, i = 1, ..n$ from S using the hit-and run algorithm.
3. For each cut $c_j \in \mathcal{C}$ compute the number of samples c_{jL} to the left of c_j and the right of $c_{jR} = n - c_{jL}$. Let $c_{jD} = |c_{jL} - c_{jR}|$.
4. Pick $c_j^* = min(c_{jD}|j = 1, ..|\mathcal{C}|)$.

Since the sampling technique provides a good estimate of volume this technique provides good cuts.

4 Computational Results

Computational experiments were performed using the interview method outlined above using some randomly generated data. We first describe the experimental setup and then provide results.

4.1 Experimental Setup

We ran the computational tests for different cases of the types of response, namely those where responses are equalities (Type-1) and those where the comparison is ordinal (Type-2). For each case, we look at the performance of the method as the total number of attributes are increased from 3 to 10. In each case, the method is started with only the basic assumptions of the feasible polyhedron, where the only initial constraints are of the form $0 \leq w_i \leq 1$, and $\sum w_i = 1$. Hence, we have 16 of these response-type-and-total-attribute cases.

For each such case, we independently sampled a thousand points from the basic feasible polyhedron, to represent a thousand samples of the actual preferences of the respondent. We then use an response oracle, which at every stage, depending on the actual preference values (weights) and the type of response being required off it, gives a consistent response. The method is now tested on these thousand samples of the actual preferences, and performance estimates are generated from these runs.

At each stage of the runs, the centroids of the feasible polyhedra, as reported by the hit-and-run sampling algorithm, forms our estimate of the true preferences. The number of samples used to compute the centroid impact the acuracy of the estimate. Based on the cubic convergence result from Lovasz [6], a sample size of 10,000 is sufficient for dimensions upto 20.

The convergence of these points to the true preference values forms one possible performance measure. Note however that a good estimate of the true weights is useful only in that it should be able to predict correctly the right order of preference of any set of bids offered to the respondent. We will therefore use a measure of the accuracy of the prediction of the order of a representative sample of bids as our main performance measure. Towards this goal, we take a sample of 10 and 20 uniformly chosen bids B_i and test how often our estimates (for each of the 1000 response-attribute cases) get the true highest preferred bid, the highest and the second highest preferred bids and so on correctly.

For the case of the equality responses (type I), we also model the case where the responses are allowed to be uncertain. By this, we mean that the responses could be of the form $\sum w_i d_i \leq c_1$ and $\sum w_i d_i \geq c_2$, where $c1 \geq c2$ and they maintain consistency, i.e., the true difference in utilities is within $[c_2, c_1]$.

Num Bids	10		20		30	
Dimn						
3	3	3	3	3	3	3
4	4	4	4	4	4	4
5	5	5	5	5	5	5
6	6	6	6	7	6	8
7	7	7	7	9	8	10
8	8	9	10	12	11	14
9	10	12	12	15	13	17
10	12	14	15	18	15	19

Table 1: Number of queries needed to get the top bid with %90 accuracy, and top two with 80% accuracy respectively - Type 1 Response with uncertainty

4.2 Results

One primary concern in the use of interview techniques is that interviewees loose interest rather quickly. In procurement settings, it is the ordinal ranking of bids that is most useful in selecting a winner. As a result the experiments presented here use the following metric: Given a set of bids and a model for the weights how many questions does the interview need to identify the top bid (or top two bids) accurately. Another important metric for interview techniques is that the time required to generate the next question should be very small since humans are notoriously impatient when it comes to waiting in front of computers.

Table 1 above presents results the number queries required to achieve 90% (80%) accuracy with respect to the top (top two) bids. The accuracy is calculated over 2000 different sample weight vectors. The number of dimensions and the number of bids (over which the top bid is being evaluated) are the two parameters that are being varied. The results in the table are encouraging in that to get 90% accuracy with respect to identifying the top bids the numner of questions is about the order of the dimensions. The rate of convergence for this case is also illustrated in Figure 2 below. Note that the responses for this case is a Type 1 response with 5% uncertainty in the

response where the bid comparisons are modeled as

$$u_{12} - \epsilon \leq \sum_i w_i(x_{1i} - x_{2i}) \leq u_{12} + \epsilon$$

where $\epsilon = 0.025$.

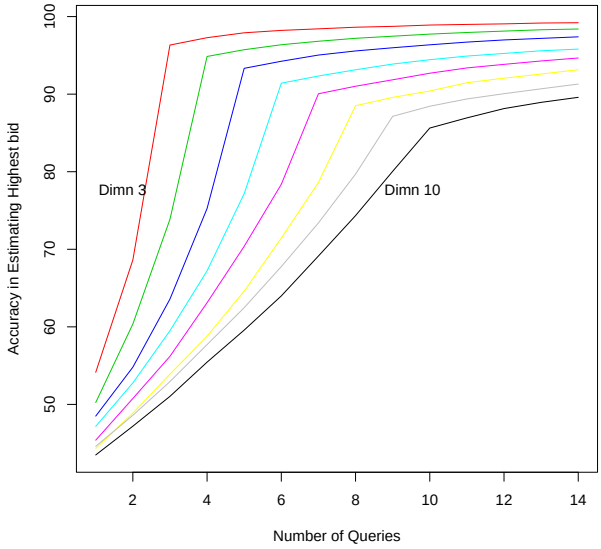

Figure 2: Convergence Rate for Top Bid with 20 Bids- Type 1 Inequality Responses

Similar results were obtained (see Figure 3) for Type-1 constraints with equalities. Note however, in this case the number of questions required is at most $N-1$ where N is the number of dimensions. Similar results were obtained for the case with Type-2 responses and the convergence rate is shown in Figure 4 below.

Table 2 presents the CPU time requires to compute the next question in each step. It is extremely important to keep this time low if we want a responsive interview tool. The CPU time is always quite small (less than a second).

5 Conclusion

We have developed an interview method that uses the hit-and-run algorithm to compute the centroid and the hyperplane that minimizes the feasible region. The at-

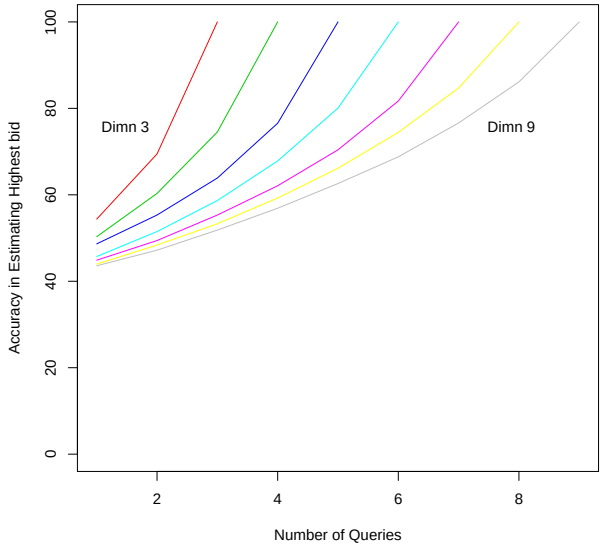

Figure 3: Convergence Rate for Top Bid with 20 Bids - Type 1 Equality Responses

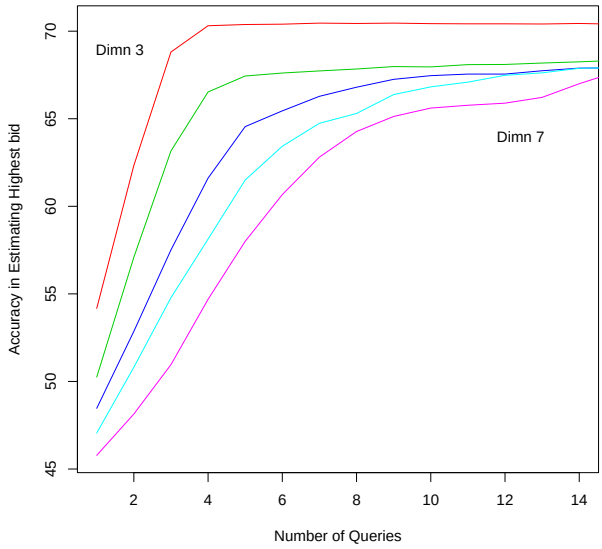

Figure 4: Convergence Rate for Top Bid with 20 Bids - Type 2 Responses

tractiveness of this approach is the relative simplicity of the approach. The core computations that are required use at most matrix algebra as compared tot he non-

Average CPU-times Per Response					
Type I		Type I uncertain		Type II	
Dimn	Cpu-time	Dimn	Cpu-time	Dimn	Cpu-time
3	0.18079	3	0.147486	3	0.0822824
4	0.20021	4	0.16798	4	0.0942722
5	0.238015	5	0.185699	5	0.107883
6	0.26438	6	0.208721	6	0.124252
7	0.288305	7	0.257243	7	0.219337
8	0.317683	8	0.284564		
9	0.352161	9	0.306698		
10	0.421903	10	0.336635		

Table 2: Average CPU times (in seconds) spent on each response-generation step of the method for various cases of responses and number of attributes (dimensions)

linear optimization techniques in the literture. The results of this method in terms of number of questions required to identify the top bid compare favorably to those in the literature and the time required (in terms of sampling) is relatively small (in the order of 1 second or less). This approach holds great promise for widespread use since it can be deployed in a standalome fashion without dependence on optimization solvers.

References

[1] Boneh, A., and A. Golan, "Constraints' Redudancy and Feasible Region Boundedness by Random Feasible Point Generator (RFPG)", In Third European Congress on Operations Research, EURO III, Amsterdam (April 9-11).

[2] Caron, R., H. Greenberg, and A. Holder, "Analytics Centers an repelling inequalitie" CCM 142, Center for Computational Mathematics, University of Colorado, Denver 1999.

[3] Dyer, M. "The complexity of vertex enumeration methods", Mathematics of Operations Research, Vol 8(3), 1983.

[4] Iyengar, V., J. Lee, M. Campbell, "Q-Eval: Evaluating Multiple Attribute Items using Queries", In Proceeding of ACM Conference on eCommerce, ACM-EC01, Jan 2001.

[5] Kaufman,D.E., and R.L. Smith, "Direction Choice for Accelerated Convergence for Hit-and-Run Sampling", Operations Research, Vol 38, 1994.

[6] Lovasz, L, "Hit-and-run mizes fast", Working paper, Microsoft Research, 2001.

[7] R.L. Smith, "Efficient Monte Carlo Procedures for Generating Random Feasible Solutions over Bounded Regions", Operations Research Vol 32, 1984.

[8] Toubia, O., D. Simester, and J.R. Hauser, "Fast Polyhedral Adaptive Conjoint Estimation", Working Paper, MIT Sloan School, May 2001.

[9] Sonnenvend, G., "An analytic centre for polyhedrons and new classes of global algorithms for linear programming", In System modeling and optimization (Budapest 1985), Berlin, Springer 1986.