# Optimization and Elicitation with the Maximin Utility Criterion

**Paolo Viappiani**[1] and **Christian Kroer**[2]

**Abstract.** We investigate robust decision-making under utility uncertainty, using the *maximin* criterion, which optimizes utility for the worst case setting. We show how it is possible to efficiently compute the maximin optimal recommendation in face of utility uncertainty, even in large configuration spaces. We then introduce a new decision criterion, *setwise maximin utility (SMMU)*, for constructing optimal recommendation sets: we develop algorithms for computing SMMU, and prove (analogously to previous results related to regret-based and Bayesian elicitation) that SMMU determines choice sets for queries that are myopically optimal. We also present experimental results showing performance of SMMU on randomly generated elicitation problems.

## 1 Introduction

Learning the preferences of the user [10] is an important problem in many domains, including decision support and recommender systems, personal agents and cognitive assistants. Because acquiring user preferences is expensive (with respect to time and cognitive cost), it is essential to provide techniques that can reason with partial preference (utility) information, and that can effectively elicit the most relevant preference information.

Following recent works in AI, we cast decision-making and elicitation as a problem of optimization under uncertainty. Adaptive utility elicitation [5] tackles the challenges posed by preference elicitation by representing the system knowledge about the user in form of *beliefs*, that are updated following user responses. Elicitation queries can be chosen adaptively given the current belief. In this way, one can often make good (or even optimal) recommendations with sparse knowledge of the user's utility function.

In this paper, we investigate the problem of producing robust recommendations using the *maximin* criterion. Maximin is the most pessimistic decision criterion; the recommended decision or option is the one associated with the highest utility in the worst case.

We examine the *strict* uncertainty setting: all we are given is a set of constraints that encode the possible utility functions (usually obtained through some form of user feedback, such as responses to elicitation queries of the type: *"Which of these products do you prefer ?"*). We argue that maximin can be adopted as a suitable robust decision criterion for decision making in the presence of such utility function uncertainty. Furthermore, we extend this idea to sets, defining the *setwise maximin* utility criterion, and we discuss the problem of interactive elicitation (which can be viewed as active preference learning). Finally, we show how linear and mixed integer programming techniques can be used to efficiently optimize both singleton recommendations and sets in large configuration spaces.

### 1.1 Assumptions

We assume a recommendation system is charged with the task of recommending an option to a user in some multiattribute space, for instance, the space of possible product configurations from some domain (e.g., computers, cars, apartment rental, etc.). Products are characterized by a finite set of attributes $\mathcal{X} = \{X_1, ...X_n\}$, each with finite domains $Dom(X_i)$. Let $\mathbf{X} \subseteq Dom(\mathcal{X})$ denote the set of *feasible configurations*. For instance, attributes may correspond to the features of various apartments, such as size, neighborhood, distance from public transportation, etc., with $\mathbf{X}$ defined either by constraints on attribute combinations (e.g., constraints on computer components that can be put together), or by an explicit database of feasible configurations (e.g., a rental database).

The user has a *utility function* $u : Dom(\mathcal{X}) \to \mathbf{R}$. In what follows we will assume either a *linear* or *additive* utility function depending on the nature of the attributes [8]. In both additive and linear models, $u$ can be decomposed as follows[3]:

$$u(\mathbf{x}) = \sum_i f_i(x_i) = \sum_i \lambda_i v_i(x_i)$$

where each *local* utility function $f_i$ assigns a value to each element of $Dom(X_i)$. In classical utility elicitation, these values can be determined by assessing local value functions $v_i$ over $Dom(X_i)$ that are normalized on the interval $[0, 1]$, and importance weights $\lambda_i$ ($\sum_i \lambda_i = 1$) for each attribute [7, 8]. This sets $f_i(x_i) = \lambda_i v_i(x_i)$ and ensures that global utility is normalized on the interval $[0, 1]$. A simple additive model in the rental domain might be:

$$u(Apt) = f_1(Size) + f_2(Distance) + f_3(Nbrhd)$$

When $Dom(X_i)$ is drawn from some real-valued set, we often assume that $v_i$ (hence $f_i$) is linear in $X_i$.[4]

We note that our framework subsumes the case of "unfactored" utilities (the utility of an option is an unknown latent value that does not factor into attributes or features); this case can be modeled by considering a parameter to represent the utility of the option.

$$u(x^i; w) = w_i \tag{1}$$

Vector $\mathbf{w} = (w_1, ..., w_n)$ is then composed of the utilities for each option. Prior knowledge can provide lower bounds and upper bounds

[1] Aalborg University, Denmark, email: paolo@cs.aau.dk
[2] Carnegie Mellon University, USA, email: ckroer@cs.cmu.edu

[3] In our notation, we use bold lowercase for vectors
[4] Our presentation relies heavily on the additive assumption, though our approach is easily generalized to more general models such as GAI [7, 4]. The assumption of linearity is simply a convenience; nothing critical depends on it.

for $w_1, ..., w_n$. $W$ is then a "hyper rectangular" region of possible utility values. Unfactored models are of limited applicability. One main drawback is that we need one utility parameter for each available option. The advantages of a factored (multi-attribute) utility representation is that preference statements, such as responses to comparison queries between two options $\mathbf{x}$ and $\mathbf{y}$, can "generalize" to other options, that have some features in common.

Since a user's utility function is not generally known, we write $u(\mathbf{x}; w)$ to emphasize the dependence of $u$ on user-specific parameters. In the additive case, the values $f_i(x_i)$ over $\cup_i\{Dom(X_i)\}$ serve as a sufficient parameterization of $u$ (for linear attributes, a more succinct representation is possible). The optimal product for the user with utility parameters $w$ is $argmax_{\mathbf{x}\in\mathbf{X}} u(\mathbf{x}; w)$. Our goal is to recommend, or help the user find, an optimal, or near optimal, product.

## 2 Decision-making with Maximin Utility

Much work in AI, decision analysis and operations research has been devoted to effective elicitation of preferences [13, 2, 6, 1, 14]. Adaptive preference elicitation generally differs from classical utility assessment in that it recognizes that good, even optimal, decisions can often be recommended with very sparse knowledge of a user's utility function [2]; and that the value of information associated with specific elicitation actions (e.g., queries)—in terms of its impact on decision quality—is often not worth the cost of obtaining it [6, 1]. This means we must often take decisions in the face of an incompletely specified utility function.

In this work, we adopt the notion of *maximin utility* as our decision criterion for robust decision making under utility function uncertainty.

Assume that through some interaction with a user, and possibly using some prior knowledge, we determine that her utility function $w$ lies in some set $W$. (The form of $W$ will become clearer when we discuss elicitation below). We define:

**Definition 1** *Given a set of feasible utility functions $W$, the* min utility (MU) $MU(\mathbf{x}; W)$ *of $\mathbf{x} \in \mathbf{X}$ is defined as:*

$$MU(\mathbf{x}; W) = \min_{w\in W} u(\mathbf{x}; w)$$

**Definition 2** *The* maximin utility $MMU(W)$ *of $W$ and the corresponding* minimax optimal configuration $\mathbf{x}^*_W$ *are defined as follows:*

$$\begin{aligned} MMU(W) = & \max_{\mathbf{x}\in\mathbf{X}} MU(\mathbf{x}; W) = & \max_{\mathbf{x}\in\mathbf{X}} \min_{w\in W} u(\mathbf{x}; w) \\ \mathbf{x}^*_W = & \arg\max_{\mathbf{x}\in\mathbf{X}} MU(\mathbf{x}; W) = & \arg\max_{\mathbf{x}\in\mathbf{X}} \min_{w\in W} u(\mathbf{x}; w) \end{aligned}$$

Intuitively, $MU(\mathbf{x}; W)$ is the worst-case utility associated with recommending configuration $\mathbf{x}$; i.e., by assuming an adversary will choose the user's utility function $\mathbf{w}$ from $W$ to minimize the utility. The maximin optimal configuration $\mathbf{x}^*_W$ is the configuration that maximizes this minimum utility. Any choice that is not maximin optimal has strictly lower utility than $\mathbf{x}^*_W$ for some $\mathbf{w} \in W$.

Maximin utility (as does minimax regret [15]) relies on relatively simple prior information in the form of bounds or constraints on user preferences (rather than probabilistic priors); and exact computation is much more tractable (in contrast with probabilistic models of utility that generally require reasoning with densities that have no closed form [1, 6]). In configuration problems, optimization over product space $\mathbf{X}$ is often formulated as a CSP or mixed integer program (MIP). In such domains, maximin utility computation can be formulated as a MIP, and solved practically for large problems using techniques such as Bender's decomposition and constraint generation [2, 4].

## 3 Optimal Recommendation Sets

In general, there is a tension between recommending the best options to the user, and acquiring informative feedback from the user. Since utility is uncertain, there is often value in recommending a *set* of options from which the user can choose her most preferred. Picking a "diverse" set of recommended options increases the odds of recommending at least one item with high utility. Intuitively, such a set of "shortlisted" recommendations should include options that are *diverse* in the following sense: recommended options should be highly preferred relative to a wide range of "likely" user utility functions (relative to the current belief) [11, 3]. This stands in contrast to some recommender systems that define diversity relative to product attributes [12], with no direct reference to beliefs about user utility. It is not hard to see that "top $k$" systems, those that present the $k$ options with highest expected utility, do not generally result in good recommendation sets [11].

Among the many possible types of queries, we focus on *choice queries*. Such queries are commonly used in conjoint analysis and product design [9], requiring a user to indicate which choice/product is most preferred from a set of $k$ options. Hence, we can view any set of products as either a recommendation set or query (or choice) set. Given a set, one can ask: what is the value of the set viewed as recommendation set; or what is its value as a query?

Recently, Viappiani and Boutilier [16, 15] showed how these two problems are connected to each other, under both a Bayesian framework or when one assumes *minimax regret* as a criterion. In the following we show the same connection when minimax utility is used as the decision criterion.

### 3.1 Setwise Maximin Utility

Suppose we have a slate of $k$ options to present to the user and want to quantify the minimum utility obtained by restricting the user's decision to options in that slate. Intuitively, the user may select any of the $k$ options as being "optimal." An adversary wanting to minimize utility should do so assuming that any such choice is possible, as we allow the user to pick *any* of the $k$ options. Formally, we choose the set of $k$ options first, but delay the specific choice from the slate until *after* the adversary has chosen a utility function $\mathbf{w}$. The maximin utility is the utility of the best option w.r.t. $\mathbf{w}$ in the slate. (To keep notation to a minimum, we assume $\mathbf{Z}$ is restricted to suitable subsets of $\mathbf{X}$ (e.g., of cardinality $k$) without making this explicit.)

**Definition 3** *Let $W$ be a feasible utility set, $\mathbf{Z} \subseteq \mathbf{X}$. Define:*

$$\begin{aligned} SMU(\mathbf{Z}, W) &= \min_{w\in W} \max_{\mathbf{x}\in\mathbf{Z}} \ u(\mathbf{x}; w) \\ SMMU(W) &= \max_{\mathbf{Z}\subseteq\mathbf{X}} \min_{w\in W} \max_{\mathbf{x}\in\mathbf{Z}} \ u(\mathbf{x}; w) \\ \mathbf{Z}^*_W &= \arg\max_{\mathbf{Z}\subseteq\mathbf{X}} \min_{w\in W} \max_{\mathbf{x}\in\mathbf{Z}} \ u(\mathbf{x}; w) \end{aligned}$$

The *setwise minimum utility(SMU)* of a set $\mathbf{Z}$ of $k$ options reflects the intuitions above. *Setwise maximin utility (SMMU)* is SMU of the minimax optimal set $\mathbf{Z}^*_W$, i.e., the set that maximizes $SMU(\mathbf{Z}, W)$.

Setwise maximin utility has some intuitive properties. First, adding new items to a recommendation set cannot decrease SMU:

**Observation 1** $SMU(\mathbf{A} \cup \mathbf{B}, W) \geq SMU(\mathbf{A}, W)$.

Incorporating options that are known to be dominated given $W$ does not change setwise maximin utility:

**Observation 2** *If $u(\mathbf{a}, w) > u(\mathbf{b}, w)$ for some $\mathbf{a} \in \mathbf{Z}$ and all $\mathbf{w} \in W$, then $SMU(\mathbf{Z} \cup \{\mathbf{b}\}, W) = SMU(\mathbf{Z}, W)$.*

**Observation 3** *MU and SMU can be explicitly expressed as the minimization over different utility spaces*

$$MU(\mathbf{A}; W_1 \cup W_2) = \min\{MU(\mathbf{A}; W_1), MU(\mathbf{A}; W_2)\}$$
$$SMU(\mathbf{A}; W_1 \cup W_2) = \min\{SMU(\mathbf{A}; W_1), SMU(\mathbf{A}; W_2)\}$$

The choice of $x \in \mathbf{Z}$ for $SMU$ is dictated by which $x$ has the highest utility with respect to the chosen $\mathbf{w} \in W$. Due to this, the different choices of $x \in \mathbf{Z}$ define a partition of the utility space, where a partition with respect to a given $x$ is the region of $W$ where the utility of $x$ is higher than any other option in $\mathbf{Z}$. We make this partition explicit:

$$W[\mathbf{Z} \to \mathbf{x}_i] = \{\mathbf{w} \in W : u(\mathbf{x}_i; w) > u(\mathbf{x}_j; w) \ \forall j \neq i, 1 \leq j \leq k\}$$

(i.e., the region of $\mathbf{w}$ where $\mathbf{x}_i$ has greater utility than any other option in $\mathbf{Z}$). The regions $W[\mathbf{Z} \to \mathbf{x}_i]$, $\mathbf{x}_i \in \mathbf{Z}$, partition $W$ (we ignore ties over full-dimensional subsets of $W$, which are easily dealt with, but complicate the presentation). We call this the $\mathbf{Z}$-*partition of* $W$. Using the $\mathbf{Z}$-partition, we can rewrite SMU:

**Observation 4** *Let $\mathbf{Z} = \{\mathbf{x}_1, \ldots, \mathbf{x}_k\}$. Then*

$$\begin{aligned} SMU(\mathbf{Z}, W) &= \min_{\mathbf{x} \in \mathbf{Z}} \min_{w \in W[\mathbf{Z} \to \mathbf{x}]} u(\mathbf{x}, w) \\ &= \min_{i=1 \leq \ldots \leq k} MU(\mathbf{x}_i, W[\mathbf{Z} \to \mathbf{x}_i]) \end{aligned}$$

We use a similar notation to express the combination of two partitions: $W[\mathbf{Z}_1 \to \mathbf{x}_i, \mathbf{Z}_2 \to \mathbf{x}_j] = W[\mathbf{Z}_1 \to \mathbf{x}_i] \cap W[\mathbf{Z}_2 \to \mathbf{x}_j]$. Using this notation, we observe the following inequality for all $i, j$: (the proof is straightforward from the definition)

**Observation 5** *For all $i, j \in \{1 \ldots k\}$:*

$$MU(\mathbf{x}_i, W[\mathbf{Z} \to \mathbf{x}_i, \mathbf{Z} \to \mathbf{x}_j] \geq MU(\mathbf{x}_i, W[\mathbf{Z} \to \mathbf{x}_i])$$

## 3.2 Optimal Myopic Elicitation

Usually, utility information is not readily available, but must be acquired through an elicitation process. Since elicitation can be costly, it is important to ask queries that elicit the most information. Our setwise maximin utility criterion can be used directly for this purpose, implementing a form of preference-based diversity. This stands in contrast to "product diversity" typically considered in recommender systems based on critiquing. And unlike recent work in polyhedral conjoint analysis [14], which emphasizes volume reduction of the utility polytope $W$, our maximin utility-based criterion is sensitive to the range of feasible products and does not reduce utility uncertainty for its own sake.

Any set $\mathbf{Z}$ can be interpreted as a query (or system-generated dynamic compound critique): We simply allow the user to state which of the $k$ elements $\mathbf{x}_i \in \mathbf{Z}$ she prefers. We refer to $\mathbf{Z}$ interchangeably as a *query* or a *choice set*. The choice of some $\mathbf{x}_i \in \mathbf{Z}$ refines the set of feasible utility functions $W$ by imposing the $k-1$ linear constraints $u(\mathbf{x}_i; \mathbf{w}) > u(\mathbf{x}_j; \mathbf{w}), j \neq i$.

When treating $\mathbf{Z}$ as a choice set (as opposed to a recommendation set), we are not interested in its maximin utility, but rather in *how much a query response will reduce maximin utility*. In our distribution-free setting, the most appropriate measure is *posterior maximin utility*, a measure of the value of information of a query. Generalizing the pairwise measure of [2], we define:

**Definition 4** *The* worst case posterior maximin utility (WP) *of $\mathbf{Z} = \{\mathbf{x}_1, \ldots, \mathbf{x}_k\}$ is*

$$WP(\mathbf{Z}, W) = \min[MMU(W[\mathbf{Z} \to \mathbf{x}_1]), \ldots, MMU(W[\mathbf{Z} \to \mathbf{x}_k])]$$

*which can be rewritten as:*

$$WP(\mathbf{Z}, W) = \min_{\mathbf{x} \in \mathbf{Z}} \max_{\mathbf{x}' \in \mathbf{X}} \min_{\mathbf{w} \in W[\mathbf{Z} \to \mathbf{x}]} u(\mathbf{x}', \mathbf{w})$$

*An* optimal choice set $OptQuery(W)$ *is any* $\mathbf{Z}$ *that maximizes worst case posterior maximin utility* $MaxWP(W)$*:*

$$MaxWP(W) = \max_{\mathbf{Z} \subseteq \mathbf{X}} WP(\mathbf{Z}, W)$$

Intuitively, each possible response $\mathbf{x}_i$ to the query $\mathbf{Z}$ gives rise to updated beliefs about the user's utility function. We use the worst-case response to measure the quality of the query (i.e., the response that leads to the updated $W$ with lowest maximin utility). The optimal query is that which maximizes this value. We observe:

**Observation 6** $WP(\mathbf{Z}, W) \geq SMU(\mathbf{Z}, W)$.

**Proof** If we consider the definition of $WP(\mathbf{Z}, W)$ and the equation for $SMU(\mathbf{Z}, W)$ in observation 4, we see that they are the same except that $WP(\mathbf{Z}, W)$ picks a maximizing $\mathbf{x}' \in \mathbf{X}$ after $\mathbf{x} \in \mathbf{Z}$ has been picked. Since $\mathbf{X}$ includes all options, $\mathbf{x}'$ can at worst be equal to $\mathbf{x}$. ■

Using this fact, we introduce a transformation that modifies a given recommendation set $\mathbf{Z}$ in such a way that SMU cannot decrease and usually increases. This will be used both for proving the optimality of SMU as a choice set, and as a heuristic for efficiently generating choice sets. Define the transformation $T$ to be a mapping that updates a given recommendation set $\mathbf{Z}$ in the following way: (a) First we construct the $\mathbf{Z}$ partition of $W$; (b) we then compute the *single recommendation* that has maximin utility in each region of the partition of $W$; (c) finally, we let $T(\mathbf{Z})$ be the new recommendation set consisting of these new recommendations.

**Definition 5** *Let $\mathbf{Z} = \{\mathbf{x}_1, \ldots, \mathbf{x}_k\}$. We define*

$$T(\mathbf{Z}) = \{\mathbf{x}^*_{W[\mathbf{Z} \to \mathbf{x}_1]}, \ldots \mathbf{x}^*_{W[\mathbf{Z} \to \mathbf{x}_k]}\}$$

Using Observation 3 and Observation 4, we prove the following.

**Observation 7** *Let $\mathbf{Z} = \{\mathbf{x}_1, \ldots, \mathbf{x}_k\}$. Let be $W^1, \ldots, W^l$ be any partition of $W$.*

$$\begin{aligned} WP(\mathbf{Z}, W) &= \min_i MMU(W[\mathbf{Z} \to \mathbf{x}_i]) \\ &= \min_i MU(\mathbf{x}^*_{W[\mathbf{Z} \to \mathbf{x}_i]}, W[\mathbf{Z} \to \mathbf{x}_i]) \\ &= \min_{i,j} \{MU(\mathbf{x}^*_{W[\mathbf{Z} \to \mathbf{x}_i]}, W[\mathbf{Z} \to \mathbf{x}_i] \cap W^j)\} \end{aligned}$$

*In particular, if we consider $T(\mathbf{Z}) = \{\mathbf{x}'_1, \ldots, \mathbf{x}'_k\}$ where $\mathbf{x}'_i = \mathbf{x}^*_{W[\mathbf{Z}\to\mathbf{x}_i]}$ and its induced partition on $W$, the exapression above become the following.*

$$WP(\mathbf{Z}, W) = \min_{i,j}\{MU(\mathbf{x}^*_{W[\mathbf{Z}\to\mathbf{x}_i]}, W[\mathbf{Z} \to \mathbf{x}_i; T(\mathbf{Z}) \to \mathbf{x}'_i]\}$$

Using this, we can now prove the following lemma:

**Lemma 1** $SMU(T(\mathbf{Z}), W) \geq WP(\mathbf{Z}, W)$

**Proof** Let $T(\mathbf{Z}) = \{\mathbf{x}'_1, \ldots, \mathbf{x}'_k\}$ where $\mathbf{x}'_i = \mathbf{x}^*_{W[\mathbf{Z}\to\mathbf{x}_i]}$. The previous observations allow to write WP and SMU compactly

$$WP(\mathbf{Z}, W) = \min_{i,j}[MU(\mathbf{x}'_i, W[\mathbf{Z} \to \mathbf{x}_i, T(\mathbf{Z}) \to \mathbf{x}'_j])] \quad (2)$$

$$SMU(T(\mathbf{Z}), W) = \min_{i,j}[MU(\mathbf{x}'_j, W[\mathbf{Z} \to \mathbf{x}_i, T(\mathbf{Z}) \to \mathbf{x}'_j])] \quad (3)$$

We now compare the two expressions componentwise. Consider the utility space $W[\mathbf{Z} \to \mathbf{x}_i, T(\mathbf{Z}) \to \mathbf{x}'_j]$: if $i = j$ then the two $MU$ components are the same. If $i \neq j$, consider any $w \in W[\mathbf{Z} \to \mathbf{x}_i, T(\mathbf{Z}) \to \mathbf{x}'_j]$. Since $w \in W[T(\mathbf{Z}) \to \mathbf{x}'_j]$, we must have $u(\mathbf{x}'_j; w) > u(\mathbf{x}'_i; w)$. Therefore $MU(\mathbf{x}'_j, W[\mathbf{Z} \to \mathbf{x}_i, T(\mathbf{Z}) \to \mathbf{x}'_j]) \geq MU(\mathbf{x}'_i, W[\mathbf{Z} \to \mathbf{x}_i, T(\mathbf{Z}) \to \mathbf{x}'_j])$. In the expression of $SMU(T(\mathbf{Z}))$ (Eq. 3), each element is no less than its correspondent in the $WP(\mathbf{Z})$ expression (Eq. 2). Thus $SMU(T(\mathbf{Z}), W) \geq WP(\mathbf{Z}, W)$. ■

From observation 6 and lemma 1 it follows that $SMU(T(\mathbf{Z}), W) \geq SMU(\mathbf{Z}, W)$.

**Theorem 1** *Let $\mathbf{Z}^*_W$ be a maximin optimal recommendation set. Then $\mathbf{Z}^*_W$ is an optimal choice set: $WP(\mathbf{Z}^*_W, W) = MaxWP(W)$.*

**Proof** Suppose $\mathbf{Z}^*_W$ is not an optimal choice set, i.e., there is some $\mathbf{Z}'$ such that $WP(\mathbf{Z}', W) > WP(\mathbf{Z}^*_W, W)$. If we apply transformation $T$ to $\mathbf{Z}'$ we obtain a set $T(\mathbf{Z}')$, and by the results above we have: $SMU(T(\mathbf{Z}', W)) \geq WP(\mathbf{Z}', W) > WP(\mathbf{Z}^*, W) \geq SMR(\mathbf{Z}^*_W, W)$. This contradicts the (setwise) maximin optimality of $\mathbf{Z}^*_W$. ■

## 4 Maximin Utility Optimization

In this section we formalize the problem of generating recommendations (both single recommendations and setwise recommendations) using mathematical programming techniques (linear programming models and mixed integer programming models).

In the following we assume the utility to be linear in $w$: $u(\mathbf{x}; w) = w \cdot \mathbf{x}$. In this case $W$ is convex polytope effectively represented by a set of constraints. Whenever the user answer a query, new constraints are added. We denote with *Constraints(W)* the set of constraints that represent the space of feasible utility functions (consistent with the user's answers).

**MU(x, W)** Given a configuration $\mathbf{x}$ and a space of possible utility functions $W$ (encoded by linear constraints), the minimum utility of $x$ can be found by solving the following linear problem ($w_i^\perp$ and $w_i^\top$ are a lower and upper bound on the values of the utility parameters $w_i$; this can be used to encode a non-probabilistic "prior" on the utility parameters).

$$\begin{aligned} \min \quad & \mathbf{w} \cdot \mathbf{x} = \sum_{1 \leq i \leq n} x_i \cdot w_i & \\ \text{s.t. } & Constraints(W) & (4) \\ & w_i^\perp \leq \mathbf{w}_i \leq w_i^\top \qquad \forall i \in \{1 \ldots n\} & (5) \end{aligned}$$

Decision variables: $\mathbf{w}$ (vector of size $n$)

**MMU(W)** Given a space of possible utility functions $W$ (encoded by linear constraints), the problem is to find the configuration $\mathbf{x}^*_W$ that is associated with maximin utility. In order to "break" the maximin optimization, we make use of Benders decomposition:

$$\begin{aligned} \max \quad & \delta & \\ \text{s.t. } & \delta \leq \mathbf{w} \cdot \mathbf{x} \qquad \forall \mathbf{w} \in GEN & (6) \end{aligned}$$

Decision variables: $\mathbf{x}$, $\delta$

In this model, $\delta$ corresponds to the *maximin utility* of the optimal recommendation $\mathbf{x}^*_W$. Constraint 6 ensures that $\delta$ is less than the utility of choice $\mathbf{x}$ for each $\mathbf{w}$. The optimization is exact when $GEN = W$ in constraint 6. However, all the constraints over $W$ need not be expressed for each of the (continuously many) $\mathbf{w} \in W$. Since maximin utility is optimal at some vertex of $W$, we only need to apply constraints for all vertices of $W$, which we denote $Vert(W)$. However, the number of vertices in $W$ can still be potentially exponential. We apply constraint generation in order to make solving the MIP much more efficient, as very few of the vertices are usually needed. This procedure works by solving a relaxed version of the problem above—the *master problem*— using only the constraints corresponding to a small subset $GEN \subset Vert(W)$. We then test whether any constraints are violated in the current solution. This is accomplished by computing the *minimum utility* of the returned solution. If $MU$ is lower than what was found in the master problem, a constraint was violated. The vertex for this constraint (corresponding to the choice $w^a$ of the adversary) is added to the master problem, tightening the MIP relaxation. The new relaxation is computed, and this process is repeated until no violated constraints exist.

Now we provide LP and MIP formulations that extend these optimization to sets.

**SMU(Z, W)** Given a set $Z$ and a space of possible utility functions $W$ the setwise minimum utility of $Z$ can be found by solving $k$ ($k$ being the cardinality of $Z$) optimization problems, in virtue of Observation 4. Considering the Z-partition of W, we compute $MU(\mathbf{x}, W[\mathbf{Z} \to \mathbf{x}])$ for each $\mathbf{x} \in \mathbf{Z}$, using the LP model shown above. We then take the (arithmetic) minimum of the results: $\min_{x\in\mathbf{Z}} MU(\mathbf{x}, W[\mathbf{Z} \to \mathbf{x}])$.

**SMMU(W)** Given utility space $W$, we can compute the maximin optimal set (of cardinality $k$) using the following MIP.

$$\begin{aligned} \max \quad & \delta & & \\ \text{s.t. } \delta \leq & \sum_{1\leq j\leq k} v_w^j & \forall \mathbf{w} \in GEN & \quad (7) \\ v_w^j \leq & \mathbf{w} \cdot \mathbf{x}^j & \forall j \leq k, w \in GEN & \quad (8) \\ v_w^j \leq & w^\top I_w^j & \forall j \leq k, w \in GEN & \quad (9) \\ \sum_{1\leq j\leq k} & I_{\mathbf{w}}^j = 1 & \forall \mathbf{w} \in GEN & \quad (10) \\ I_{\mathbf{w}}^j \in & \{0, 1\} & \forall j \leq k, \mathbf{w} \in GEN & \end{aligned}$$

Decision variables: $\mathbf{x}^j$, $\delta$, $\mathbf{I}_\mathbf{w}$, $\mathbf{v}_w$

In this model, $\delta$ corresponds to the *setwise maximin utility* of the optimal set $\mathbf{Z}^*_W$. $M$ is an arbitrary large number; $w^\top$ is some upper bound on the values taken by the weight parameters. Constraints 7, 8 and 9 ensures that $\delta$ is less than the utility of the best option in $\{\mathbf{x}^1, ..., \mathbf{x}^k\}$ for each $\mathbf{w}$, by introducing a variable $v$ (for each $w$ and each element of the set) to represent the value of minimum utility for the item selected, and indicators $\mathbf{I}_w$ to represent the selection. Only one $\mathbf{v}_w$ will be different from zero for each $w$, and since the objective function is maximized, the optimization will set $v_w^j = \mathbf{w} \cdot \mathbf{x}^j$ for the $j$ such that $I_w^j = 1$; constraint 9 enforces 0 in the other cases.

Constraint 10 ensures that only one of the $k$ items is selected for each utility function $w$.

We employ constraint generation in a way analogous to the single item case. At each step of the optimization, we compute the *setwise minimum utility*, solved using a series of LPs (as discussed above).

**Alternative Heuristics** Setwise optimization requires solving a large number of MIPs using constraint generation strategies. We also present a number of heuristic strategies that are computationally less demanding.

- The *current solution strategy* (CSS) proceeds as follows. Consider $\mathbf{w}^a$, the adversary's utility minimizing the utility of $\mathbf{x}^*_W$, the current maximin optimal recommendation; $u(\mathbf{x}^*_W; \mathbf{w}^a) = MU(\mathbf{x}^*_W; W)$. Let's further consider $\mathbf{x}^a = \arg\max_{\mathbf{x}\in\mathbf{X}} u(\mathbf{x}; w^a)$. CSS will return the set $\mathbf{Z}_{CSS} = \{\mathbf{x}^*_W, \mathbf{x}^a\}$. We extend this to sets with cardinality greater than two. Considering a set $\mathbf{Z}$, define $\mathbf{w}^a(\mathbf{Z}) = \arg\min_{\mathbf{w}\in W} \max_{\mathbf{x}\in\mathbf{Z}} u(\mathbf{x}; w)$ and be $\mathbf{x}^a(\mathbf{Z}) = \arg\max_{\mathbf{x}\in\mathbf{X}} u(\mathbf{x}; w^a(\mathbf{Z}))$. The *chain of adversaries* strategy constructs a set of size $k$ starting by initializing $\mathbf{Z}$ to be $\mathbf{Z}_{CSS}$, the set of size two returned by the current solution strategy, and then iteratively add one element ($k - 2$ times) by setting $\mathbf{Z} := \mathbf{Z} \cup \mathbf{x}^a(\mathbf{Z})$.
- The *query iteration strategy* (QIS) directly applies the T operator until a fixed point is reached. A fixed point is such that $SMU(T(\mathbf{Z}); W) = SMU(\mathbf{Z}; W)$.

## 5 Experiments

Using randomly generated elicitation data we ran a number of experiments using the algorithms described above. For all experiments, we generated constraints on the possible options using random binary constraints of the form $\neg f_1 \vee \neg f_2$ where $f_1$ and $f_2$ are features. We also assume some prior knowledge of user preferences, represented by random utility constraints of the form $\mathbf{w} \cdot \mathbf{x_k} \geq \mathbf{w} \cdot \mathbf{x_l}$, where $\mathbf{x_k}$ and $\mathbf{x_l}$ are random assignments $\in [0,1]^m$ (not necessarily feasible options) sampled with uniform probability over all possible assignments. The user's preference values $w_1 \dots w_n$ are random and normalized such that $\sum_{w\in W} w = 1$. Finally, for all experiments we use recommendation/query sets of size ($k$) 3.

First, we ran experiments to determine how the runtime of the algorithms are affected by increasing instance sizes. This was done by running the algorithms on instances ranging from 10 to 15 features, with 30 experiments performed on each size. The average runtimes for these experiments can be seen in figure 1. As seen in the figure, runtime of exact SMMU computation becomes rapidly higher, and we were unable to perform experiments with more than 15 features, as several of the 30 experiments per size would time out with 16 features. In contrast to this we see that the runtime of the CSS and QIS algorithms do not rise significantly as the number of features increase. Due to this, we focus on the performance of CSS and QIS in the following experiments, as SMMU computation is too slow for practical use.

Using the CSS and QIS algorithms, we ran experiments to determine how the MMU optimal recommendation improves as more queries are asked. These were performed using larger instances, with 30 features per instance, 40 binary feature constraints and 40 utility constraints. In figure 2 we present the utility loss from recommending the MMU optimal recommendation as opposed to the optimal recommendation according to the user's preferences, as a function of the number of queries. The CSS and QIS algorithms have comparable performance, both improving utility loss by a small margin.

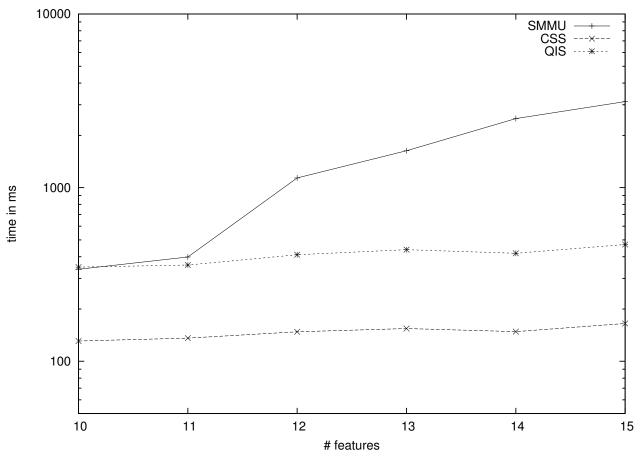

**Figure 1.** Average runtime of query computation for an increasing number of features. Averaged over 30 instances per size, with k = 3

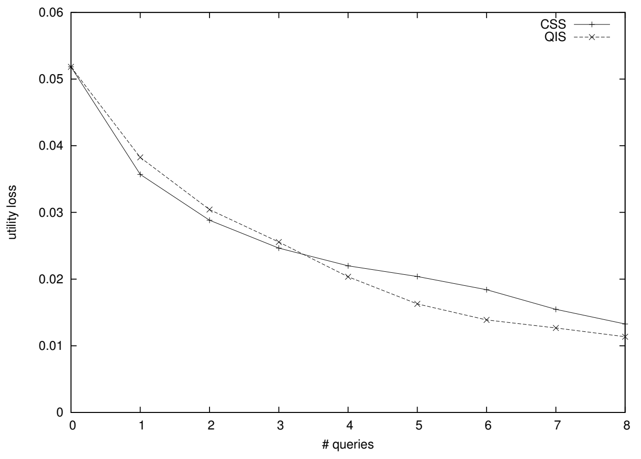

**Figure 2.** Average utility loss of the optimal recommendation as a function of the number of queries, using the CSS and QIS algorithms. Averaged over 30 instances, with k = 3.

In figure 3 we show the minimum utility guarantee from the MMU recommendation as it increases with more queries asked. It quickly increases with the first 4-5 queries, but after that there is little improvement. While our theoretical results show that there is a connection between the problem of generating recommendations and queries, our results show that the pessimistic maximin decision criterion is generally not able to effetively elicit user preferences beyond the first few queries. In this case, it might be useful to adopt a non-myopic approach, or an alternative decision criterion.

Further investigation is required to determine in which settings our framework can be used effectively in interactive elicitation, and how to avoid stalling.

We also note that it is of course possible to use maximin as a decision criterion, while resorting to other strategies (perhaps based on regret or on probabilistic methods) to decide the next query.

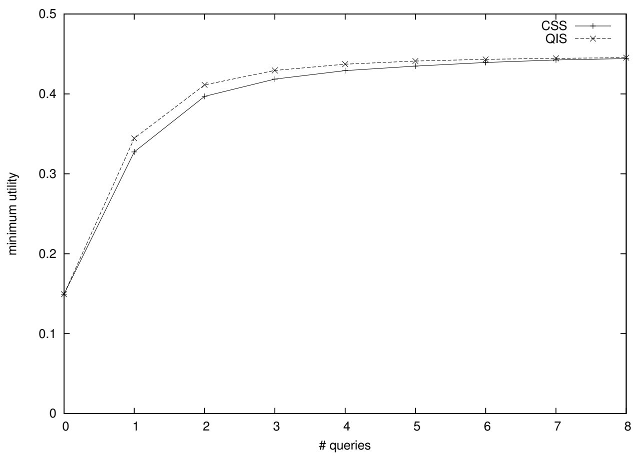

**Figure 3.** Average minimum utility as a function of the number of queries using the CSS and QIS algorithms. Averaged over 30 instances, with k = 3.

## 6 Discussion

In this paper we have developed a novel formalization for decision-making under utility uncertainty (making recommendations) using the maximin utility criterion. This approach allows for the highest degree of robustness, as the option recommended is guaranteed to ensure highest utility in the worst case. We formulated the problem of generating recommendation sets and introduced a new decision criteria. We developed computational MIP methods for optimal recommendation sets, as well as tractable approximations.

Moreover, following analogous models available for the minimax regret and Bayesian frameworks, we showed the connection between the problem of generating optimal recommendation sets and myopically optimal elicitation queries. This shows that our setwise maximin criterion, a natural extension of maximin to sets, in addition to providing robust recommendation sets, also serves as a means of generating myopically optimal choice queries (asking the user to pick his most preferred option in a slate).

Finally, we provided preliminary experimental results, showing performance of our approach on randomly generated data. We showed that maximin as an elicitation framework can provide good initial queries, but in an interactive setting it often stalls before finding the optimal recommendation.

We conclude with a remark about the choice of the decision criterion. A common criticism about maximin is that it can be overly pessimistic. Indeed expected utility (assuming a prior is available) or minimax regret may yield better recommendations in many cases. However, when a decision maker wishes guarantees on the worstcase performance (perhaps in critical decisions with high stakes), she must be willing to sacrifice "average" utility for such guarantee. This is the price to pay for the (strong) worstcase guarantees of maximin! We argue that the question of what criterion to use is almost philosophical, as there is no "right" or "wrong" decision criterion (each one might be better suited to different decision contexts).

## REFERENCES

[1] Craig Boutilier. A POMDP formulation of preference elicitation problems. In *Proc. of AAAI-02*, pages 239–246, Edmonton, 2002.

[2] Craig Boutilier, Relu Patrascu, Pascal Poupart, and Dale Schuurmans. Constraint-based optimization and utility elicitation using the minimax decision criterion. *Artifical Intelligence*, 170(8–9):686–713, 2006.

[3] Craig Boutilier, Richard S. Zemel, and Benjamin Marlin. Active collaborative filtering. In *Proc. of UAI-03*, pages 98–106, Acapulco, 2003.

[4] Darius Braziunas and Craig Boutilier. Minimax regret-based elicitation of generalized additive utilities. In *Proc. of UAI-07*, pages 25–32, Vancouver, 2007.

[5] Darius Braziunas and Craig Boutilier. Elicitation of factored utilities. *AI Magazine*, 29(4):79–92, 2008.

[6] Urszula Chajewska, Daphne Koller, and Ronald Parr. Making rational decisions using adaptive utility elicitation. In *Proc. of AAAI-2000*, pages 363–369, Austin, TX, 2000.

[7] Peter C. Fishburn. Interdependence and additivity in multivariate, unidimensional expected utility theory. *International Economic Review*, 8:335–342, 1967.

[8] Ralph L. Keeney and Howard Raiffa. *Decisions with Multiple Objectives: Preferences and Value Trade-offs*. Wiley, New York, 1976.

[9] Jordan J. Louviere, David A. Hensher, and Joffre D. Swait. *Stated Choice Methods: Analysis and Application*. Cambridge University Press, Cambridge, 2000.

[10] Bart Peintner, Paolo Viappiani, and Neil Yorke-Smith. Preferences in interactive systems: Technical challenges and case studies. *AI Magazine*, 29(4):13–24, 2008.

[11] Robert Price and Paul R. Messinger. Optimal recommendation sets: Covering uncertainty over user preferences. In *Proc. of AAAI-05*, pages 541–548, 2005.

[12] James Reilly, Kevin McCarthy, Lorraine McGinty, and Barry Smyth. Incremental critiquing. *Knowledge-Based Systems*, 18(4–5):143–151, 2005.

[13] Ahti Salo and Raimo P. Hämäläinen. Preference ratios in multiattribute evaluation (PRIME)–elicitation and decision procedures under incomplete information. *IEEE Trans. on Systems, Man and Cybernetics*, 31(6):533–545, 2001.

[14] Olivier Toubia, John Hauser, and Duncan Simester. Polyhedral methods for adaptive choice-based conjoint analysis. (4285-03), 2003.

[15] Paolo Viappiani and Craig Boutilier. Regret-based optimal recommendation sets in conversational recommender systems. In *Proceedings of the 3rd ACM Conference on Recommender Systems (RecSys09)*, pages 101–108, New York, 2009.

[16] Paolo Viappiani and Craig Boutilier. Optimal bayesian recommendation sets and myopically optimal choice query sets. In *Advances in Neural Information Processing Systems 23 (NIPS)*, Vancouver, 2010.