A Study in Preference Elicitation under Uncertainty

by

Greg Hines

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Computer Science

Waterloo, Ontario, Canada, 2011

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

In many areas of Artificial Intelligence (AI), we are interested in helping people make better decisions. This help can result in two advantages. First, computers can process large amounts of data and perform quick calculations, leading to better decisions. Second, if a user does not have to think about some decisions, they have more time to focus on other things they find important. Since users' preferences are private, in order to make intelligent decisions, we need to elicit an accurate model of the users' preferences for different outcomes. We are specifically interested in outcomes involving a degree of risk or uncertainty.

A common goal in AI preference elicitation is minimizing regret, or loss of utility. We are often interested in minimax regret, or minimizing the worst-case regret. This thesis examines three important aspects of preference elicitation and minimax regret. First, the standard elicitation process in AI assumes users' preferences follow the axioms of Expected Utility Theory (EUT). However, there is strong evidence from psychology that people may systematically deviate from EUT. Cumulative prospect theory (CPT) is an alternative model to expected utility theory which has been shown empirically to better explain humans' decision-making in risky settings. We show that the standard elicitation process can be incompatible with CPT. We develop a new elicitation process that is compatible with both CPT and minimax regret. Second, since minimax regret focuses on the worst-case regret, minimax regret is often an overly cautious estimate of the actual regret. As a result, using minimax regret can often create an unnecessarily long elicitation process. We create a new measure of regret that can be a more accurate estimate of the actual regret. Our measurement of regret is especially well suited for eliciting preferences from multiple users. Finally, we examine issues of multiattribute preferences. Multiattribute preferences provide a natural way for people to reason about preferences. Unfortunately, in the worst-case, the complexity of a user's preferences grows exponentially with respect to the number of attributes. Several models have been proposed to help create compact representations of multiattribute preferences. We compare both the worst-case and average-case relative compactness.

Acknowledgements

I would first like to thank my supervisor Kate Larson. I would not be here today if it were not for her support, guidance and trust in me. As I finish my PhD, I am privileged to be on a path as exciting as the one I am on. Prof. Larson played a key role in getting me on this path. I would also like to thank Robin Cohen and Pascal Poupart. As members of committee they provided feedback and support from the beginning. The internal-external member of my committee Selcuk Onay was able to provide key feedback from a different perspective on my research. My external committee member, David Parkes also provided valuable insight.

I would also like to thank Craig Boutilier at the University of Toronto. Prof. Boutilier's interest in my work was a major motivation. Despite being a senior faculty member with many responsibilities, Prof. Boutilier took considerable time out of his day to talk to me about both my current research and to suggest several exciting areas for possible future research.

Doing research in a complete void with no outside contact is not worth doing. My friends played a vital role in getting me through all of this. The thought of being so far away from my friends in the coming years is breaking my heart.

Finally, I would like to thank my family; both old and new. To my parents; I love you so much.

Dedication

To Lisa:

You are the best part about me

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

A common goal of Artificial Intelligence (AI) is to have computers work with people to make better decisions. A computer could help plan a trip, for example, or help someone decide on a risky medical procedure [20, 68]. A computer could also help control a household's energy usage, or create an optimal policy for a Markov Decision Process [30, 52, 64]. Having a computer make these decisions can free up a person's time. A computer can process large amounts of information. Using a computer also minimizes the risk of trivial mistakes such as addition error. In cases where a decision has to be made repeatedly, a computer will never get bored.

In all of the example decisions mentioned above there can be a significant element of risk. When planning a trip, there is always the risk of a flight being delayed or cancelled. Similarly, almost all medical procedures carry some risk to them. If we pay more for electricity when demand is high, then we need to decide if it is worthwhile waiting until the morning to run the dishwasher. Hence, this thesis is focused on making decisions where there is an element of risk.

For computers to make (or suggest) intelligent decisions, the computer must have an accurate model of the users' preferences [22]. If the user strongly dislikes missed plane connections, for example, then the computer should not suggest a flight that arrives only shortly before the connection leaves. Unfortunately, preferences are private; all we know about a person's preferences is what they choose to tell us. As a result, models of preferences will rarely be perfectly accurate. The focus of this thesis is how to create "accurate enough" models of preferences.

The basic framework for building models of preferences is through *preference elicitation*, which is based on the process of asking the user questions and using their responses to infer

constraints on their preferences. The questions are usually along the lines of, "how much more do you prefer this outcome over this other outcome?" [36] We then determine if our accumulated set of constraints gives us an accurate enough model to suggest which decision the user should choose. If not, we repeat by asking another question. If the model is accurate, we then choose a decision based on the model.

There are many different challenges with this approach. The first problem is that we must understand how people reason about preferences. This understanding is necessary to make correct inferences from the user's response. The standard model in AI preference elicitation is that people are *economically rational* [54]. However, there has been considerable work done in psychology and behavioural economics which shows that people are not economically rational [55]. This work has also produced many alternative models. Current research has not examined how compatible AI preference elicitation research is with these alternative models.

The second issue is determining if a model is accurate enough. A common measurement of accuracy is regret: the loss in utility from choosing one decision over another. If the maximum regret from choosing a decision is zero, we know that the decision is optimal. Since our models of preferences are not completely accurate, we can only estimate the regret from choosing a decision. One estimate of regret is *minimax regret* which focuses on bounding the best worst-case regret [10, 68]. Another estimate is *expected regret* which focuses on the average regret [20]. In either case, if the estimated regret for all decisions is too high, we need to improve the accuracy of the model. Since improving the accuracy means burdening the user with more questions, it is important that our estimate of regret is as accurate as possible. Minimax regret is easy to calculate but often overestimates the actual regret, sometimes by as much as an order of magnitude. Expected regret gives a more accurate estimate but requires us to make additional assumptions about what we know about the user. Sometimes these additional assumptions are reasonable and sometimes they are not.

The third problem is the issue of attributes and tractability. Attributes are a natural way of describing outcomes; for example we can describe a flight by the airline, class, *etc.* Using attributes to describe different outcomes can make it easier for the user to compare outcomes [14]. In the worst case, however, the number of outcomes will grow exponentially with respect to the number of attributes. As a result, several *multiattribute utility independence* (MUI) models have been proposed for compactly representing preferences over multiple attributes and avoiding the exponential growth [11, 25, 27, 36]. To our knowledge, there has been little comparison of how efficiently different MUI models are able to represent different multiattribute utility functions. There would be considerable benefit to developing a database of MUI models for which we can compare the relative compactness

of any two models in this set, *e.g.* to know that in the average case or worst case, model A can be exponentially more compact than model B. This would help in choosing the best MUI model for a given scenario.

1.1 Contributions

This thesis addresses each of the above three problems. In brief, our three contributions are as follows. First, we provide a new preference elicitation technique which can be used in real world settings where the standard technique cannot be used. Secondly, we greatly improve the efficiency of the elicitation process by providing a more accurate way of measuring regret. Finally, we provide a comprehensive comparison between several different MUI models which is vital when needing to choose which model to use. A more detailed discussion of these three contributions is given below.

First, we show that the standard AI preference elicitation approach is not compatible with the models of human preferences proposed by psychologists and behavioural economists. We then introduce a new preference elicitation approach, the *Gamble Equivalence Method* (GEM), which is compatible with these models. We also show that GEM can be used to efficiently reduce the minimax regret.

Second, we introduce two new methods for estimating regret: *learnt Probabilistic Minimax Regret* (lPrMMR) and *hypothesis Probabilistic Minimax Regret* (hPrMMR). As with minimax regret, lPrMMR and hPrMMR aim to give an upper bound on the actual regret. However, these bounds are probabilistic, *i.e.* there is some probability that the bounds are actually less than the actual regret. The controller (whoever is running the elicitation process) is able to choose the probability of this error. With a low probability of error both lPrMMR and hPrMMR behave similarly to minimax regret. With a higher probability of error, both lPrMMR and hPrMMR can provide a lower bound on the regret than MMR provides. The advantage is that neither lPrMMR nor hPrMMR require the strong assumptions that expected regret requires. Additionally, both lPrMMR and hPrMMR are especially well adapted to processing a series of users. For example, this means that both lPrMMR and hPrMMR could be used by websites helping people choose the best products.

Finally, we present a comparison of the compactness and abilities of several different MUI models. A common MUI model for use with preference elicitation is the *generalized additive independence* (GAI) model [27]. Braziunas and Boutilier showed how to calculate minimax regret in polynomial time with respect to the size of the GAI representation [10, 13]. Our work focuses on examining *conditional difference independence* (CDI_r), a new

MUI model [11]. While we show that minimax regret can also be calculated in polynomial time with respect to the size of the CDI_r representaion, we show that, in the worst case, the GAI model is exponentially more compact than the CDI_r model. We then show that, in the worst case, the CDI_r model is exponentially more compact than the *conditionally additive independence* model [25]. Finally, we use simulations to examine the average-case relative compactness of the GAI and CDI_r models. We are able to show that even in the average-case scenarios, there can be a considerable difference in compactness between GAI and CDI_r.

1.2 Guide to the Thesis

Chapter 2: This chapter provides the background on models of utility and preferences, preference elicitation, and multiattribute utility independence models.

Chapter 3: This chapter discusses why current AI preference elicitation techniques are incompatible with descriptive models of preferences. We also introduce GEM and show that it is both compatible with the descriptive models and can be used to efficiently reduce the minimax regret. Experimental results are included.

Chapter 4: This chapter introduces both lPrMMR and hPrMMR and compares them against minimax regret and expected regret. We also discuss different methods for optimizing both lPrMMR and hPrMMR. Experimental results are included.

Chapter 5: In this chapter, we compare how efficient different MUI models are. The comparisons include worse case analysis and experimental average case analysis. We also discuss how some MUI models may be used to efficiently calculate minimax regret.

Chapter 6: Conclusion; we also discuss areas for future research.

Chapter 2

Background

Our work focuses on eliciting the preferences from users to help make better decisions on their behalf. In Section 2.1, we review possible models for a user's preferences. We focus on *expected utility theory* and *cumulative prospect theory*; two common models for preferences in risky settings. In Section 2.2, we discuss different techniques for eliciting preferences from users. Finally, in Section 2.3, we discuss how multiattribute preferences can be modeled.

2.1 Models of Users' Preferences

Consider a set of possible outcomes $\mathbb{X} = [x_\perp, \ldots, x_\top]$ where $x_\perp$ is the least preferred outcome and $x_\top$ is the most preferred outcome. The set $\mathbb{X}$ can be continuous or discrete. We can describe a user by their preferences over $\mathbb{X}$. Let $\precsim$ be the preference operator, *i.e.* $x_1 \precsim x_2$ means that x_2 is preferred at least as much as x_1. Two standard assumptions about $\precsim$ is that it satisfies both *completeness* and *transitivity* [41, 54].

Definition 1 (Completeness [41])**.** *Given any two outcomes x and x' in $\mathbb{X}$, a user will either prefer one outcome over another or be indifferent.*

Completeness means that we can compare any two outcomes. This excludes the possibility of scenarios where we have incomparable outcomes, *e.g.* settings where one outcome is getting a toothbrush and another is doing long division.

Definition 2 (Transitivity [41])**.** *Given any three outcomes x_1, x_2, and x_3, if $x_1 \precsim x_2$ and $x_2 \precsim x_3$ then $x_1 \precsim x_3$.*

Transitivity is assumed in order to remove the possibility of "absurd" preferences. For example, suppose we have outcomes $\{x_\perp, x_1, x_\top\}$ and a user with preferences that violate transitivity, *e.g.* $x_\top \succ x_1$, $x_1 \succ x_\perp$ and $x_\perp \succ x_\top$. As a result, the user might be willing to pay \$1 to change the outcome from $x_\perp$ to x_1, from x_1 to $x_\top$ and from $x_\top$ to $x_\perp$. If we combine these three steps, the user would be willing to pay \$3 to go from $x_\perp$ to $x_\perp$. This siuation is known as a *money pump* where we can get money for doing nothing [54].

If $\precsim$ is both complete and transitive, then there exists a function $u : \mathbb{X} \to \mathbb{R}$ which represents $\precsim$, *i.e.*

$$u(x) < u(x')$$

if and only if $x \prec x'$ [41]. The function u may only imply a qualitative relationship where $u(x) > u(x')$ implies that x is preferred over x' but we cannot say by how much. We can also have a quantitative relationship, *e.g.* $u(x) = 2u(x')$ means that x is preferred twice as much as x'.

Lemma 1. *The function u is unique up to* positive affine transformation*; this means that we can create a new utility function*

$$u_1(x) = a \cdot u(x) + b$$

for any $a \in \mathbb{R}^+$ and $b \in \mathbb{R}$ and $u_1(x)$ and $u(x)$ will preserve the preferences.

Proof. Suppose we have two outcomes x and x' such that $x \prec x'$, then

$$\begin{aligned} u(x) &< u(x'), \\ au(x) &< au(x'), \\ au(x) + b &< au(x') + b, \\ u_1(x) &< u_1(x'). \end{aligned}$$

□

As a result, we typically normalize utility values so that $u(x_\perp) = 0$ and $u(x_\top) = 1$. This normalization is needed when using standard gamble queries, as discussed in Section 2.2. In many, but not all cases, we can assume *monotonicity* where $u(x_i) \geq u(x_j)$ for all $i > j$. The set of all possible utility values is $\mathbb{U} = [0, 1]^{|\mathbb{X}|}$ (assuming normalized utility values).

2.1.1 Preferences over Risky Outcomes

A *gamble* d is represented by a probability distribution over $\mathbb{X}$, *i.e.* $\Pr_d(x)$ is the probability of the outcome $x \in \mathbb{X}$ occurring as the result of the gamble d.[1] A gamble is typically written in the form

$$[p_1; x_\perp, \ldots, p_j; x_j \ldots, p_n; x_\top].$$

An example of a simple gamble is flipping a fair coin; there is a 50% probability of heads and 50% probability of tails. This gamble would be represented by $[0.5; heads, 0.5; tails]$. When there are multiple gambles and the user must choose one of them, we refer to the gambles as *decisions*.

We are interested in scenarios where the user can choose between different decisions. The set of all possible decisions for a specific scenario is $\mathbb{D}$. As with outcomes, people have preferences over decisions. These preferences are represented by $\prec_{\mathbb{D}}$, *e.g.* $d \sim d'$ means that the user is indifferent between decisions d and d' and $d \prec d'$ means that the user strictly prefers decision d' over d. We assume that $\prec_{\mathbb{D}}$ is *continuous*. To define continuity, we first define a *compound decision*.

Definition 3 (Compound decision)**.** *The compound decision*

$$[p; D, 1-p; D'],$$

means that with probability p the decision D happens and otherwise, with probability $1-p$, the decision D' happens.

Definition 4 (Continuous [41])**.** *The preference function $\succeq$ is continuous if, given decisions d_1,d_2 and d_3, the sets*

$$\{p \in [0,1] : [p; d_1, 1-p; d_2] \succeq d_3\}$$

and

$$\{p \in [0,1] : d_3 \succeq [p; d_1, 1-p; d_2]\}$$

are both closed.

Continuity means that a user's preferences do not experience "sudden" changes. An example of a non-continuous preference function over $\mathbb{D}$ where $\mathbb{X} = \{x_\perp, x_1, x_\top\}$ is

$$d \succ d' \text{ if } \begin{cases} \Pr_d(x_\top) > \Pr_{d'}(x_\top) & \text{or} \\ (\Pr_d(x_\top) = \Pr_{d'}(x_\top)) \text{ and } \Pr_d(x_1) > \Pr_{d'}(x_1). \end{cases}$$

[1] A gamble can also be called a lottery or prospect.

As long as $\prec_{\mathbb{D}}$ is continuous, there exists a function $U : \mathbb{D} \to \mathbb{R}$ such that $U(d) > U(d')$ if and only if $d \succ_{\mathbb{D}} d'$ [41]. There are many different models for U. Models can be *descriptive*, *prescriptive* or both. A descriptive model attempts to explain how people actually reason about preferences. A prescriptive model attempts to explain how people should reason about preferences. In this section, we review *expected utility theory*, the standard prescriptive model, and *cumulative prospect theory*, the standard descriptive model.

Expected Utility Theory

Expected utility theory (EUT) is based on the *von Neumann-Morgenstern expected utility function*.

Definition 5 (von Neumann-Morgenstern expected utility function [63])**.** *The utility function U is a von Neumann-Morgenstern expected utility function, if for any decision d and utility function $u : \mathbb{X} \to \mathbb{R}$,*

$$U(d, u) = \sum_{x \in \mathbb{X}} \Pr_d(x) u(x).$$

In this case, we denote the user's utility by $EU(d, u)$.

The existence of an expected utility function is dependent on *decomposability* and the *independence axiom*.

Definition 6 (Sequential decision)**.** *The sequential decision,*

$$< D, D' >,$$

means that first the decision D happens and then the decision D' happens.

Definition 7 (Decomposability [54])**.** *Any compound or sequential decision can be converted to a "basic decision."*

For example, suppose we have the compound decision $D = [0.5; \$1, 0.5; \$5]$ and $D' = [0.5; \$3, 0.5; \$6]$. Then the decision $< D, D' >$ can be reduced to $[0.25; \$4, 0.25; \$7, 0.25; \$8, 0.25; \$11]$.

Definition 8 (Independence axiom [63])**.** *The utility function U satisfies the independence axiom if and only if, for any three decisions d, d', and d'' such that $U(d) \geq U(d')$,*

$$U([\alpha; d, (1-\alpha)d'']) \geq U([\alpha; d', (1-\alpha)d'']) \ \forall \ \alpha \in [0, 1]. \tag{2.1}$$

Choice A	Choice B
\$4000 with probability of 0.8 nothing otherwise	guaranteed \$3000

(a) First choice

Choice C	Choice D
\$4000 with probability of 0.2 nothing otherwise	\$3000 with a probability of 0.25 nothing otherwise

(b) Second choice

Table 2.1: *An experiment demonstrating the problems with expected utility theory as a descriptive theory. In the first half of the experiment, participants were asked to choose between choices A and B. In the second half, participants were asked to choose between choices C and D.*

In other words, the independence axiom implies that a user's preferences between two decisions is unaffected by any third choice.

EUT says that U is an expected utility function if and only if decomposition and the independence axiom hold [41, 63].

When faced with a set of possible decisions $\mathbb{D}$ to choose between, the *Maximum Expected Utility* (MEU) principle says we should choose the decision which maximizes our expected utility [54]. In other words, we should choose the decision

$$d^* = \arg\max_{d \in \mathbb{D}} EU(d, u).$$

The reasoning behind MEU is that choosing d^* will, on average, maximize u. It is generally accepted that maximizing EU is *economically rational* [54, 55]. As a result, EU is seen as a prescriptive model of how people should reason about preferences.

Cumulative Prospect Theory

Unfortunately, expected utility is a poor descriptive model of peoples' actual preferences. Starting in the 1950s, numerous experiments have shown peoples' preferences to contradict those predicted by EU, including work by Allais and the following experiment conducted

by Kahneman and Tversky [6, 33]. In the first half of the study, people were asked to choose between the two gambles shown in Table 2.1(a). For example, if they chose A, they would receive \$4,000 with a probability of 80%, and otherwise, they would receive nothing. If they chose B, they would receive a guaranteed \$3,000. 80% of the participants chose B. According to EU, this reesult implies that

$$\begin{aligned} 0.8u(\$4,000) + 0.2u(\$0) &< u(\$3,000), \\ 0.8u(\$4,000) &< u(\$3,000). \end{aligned} \tag{2.2}$$

In the second half of the study, participants were asked to choose between the two choices in Figure 2.1(b). If participants chose C, they would get \$4,000 with a probability of 20% and otherwise nothing. With choice D, participants would get \$3,000 with a probability of 25% and otherwise nothing. This time, 65% chose C. According to EU, this results implies that

$$\begin{aligned} 0.2u(\$4,000) &> 0.25u(\$3,000), \\ 0.8u(\$4,000) &> u(\$3,000). \end{aligned} \tag{2.3}$$

Equations 2.2 and 2.3 contradict each other, regardless of the utility for the individual outcomes. Therefore, the reversal in preferences contradicts expected utility. Specifically, if we let d'' be the outcome where the user wins nothing and $\alpha = 0.25$, these results violate the axiom of independence.

This experiment has been replicated numerous times, including with non-monetary outcomes [33, 55]. Other, more fundamental, problems with expected utility have been studied. For example, people treat gambles involving only gains differently than a gamble involving losses, even if the end result is the same [33].

As a result, numerous alternative models of preferences have been proposed [55]. These models may be thought of as a series of steps in relaxing the assumptions EUT relies on. Arguably the most successful model at predicting peoples' actual preferences is *cumulative prospect theory* (CPT) [55, 59]. Daniel Kahneman won the Nobel Memorial Prize in Economics for his work on CPT, and the papers on *prospect theory* (the precursor to CPT) and CPT are two of the mostly highly cited papers in economics [31].

The three foundations of CPT are *loss aversion*, *framing*, and *cumulative probability weighting*.

Loss aversion: People are more sensitive to a loss than to a gain of the same magnitude [32]. One way of modeling loss aversion is with the utility function

$$u(x) = \begin{cases} x^\alpha & \text{if } x \geq 0 \\ -(-x^\beta) & \text{otherwise,} \end{cases}$$

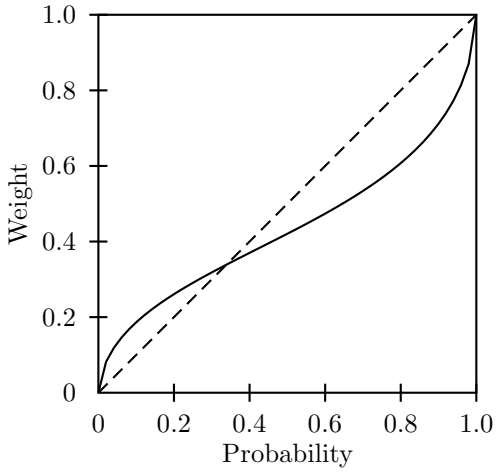


Figure 2.1: *The commonly accepted average shape of the probability weighting function* $w(p)$.

for $\beta > \alpha$. By itself, loss aversion is compatible with EU.

Framing: People pay attention to how we arrive at a final outcome, not just the final outcome itself. For example, suppose we have a disease which risks killing 1000 people [34]. One possible treatment will cure everyone with a probability of 50% and otherwise only cure 500 people. A second treatment will cure everyone but there is a 50% probability that its side effects will kill 500 people. Experiments have shown that people do not view these two treatments as equivalent. People focus on how each treatment works, not just the final number of people saved. The first treatment is framed in terms of gain while the second treatment is framed in terms of loss. This violates decomposability and therefore, framing is not compatible with EU.

Cumulative probability weighting: People distort the probabilities of events. We attach too much weight to an unlikely event and too little weight to an almost certain event. A (slightly morbid) example of probability weighting occurs when playing Russian Roulette. Suppose we have an option of paying for one bullet to be removed. Most people would pay more to go from one bullet to none than from 4 to 3 bullets [71].

The generally accepted average shape of the probability weighting function $w(p)$ is shown in Figure 2.1 [71]. The dashed line corresponds to a neutral weighting. For low probabilities $w(p)$ is above the neutral weighting and for high probabilities $w(p)$

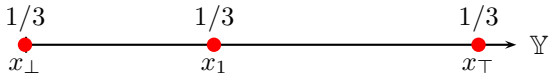


Figure 2.2: *An example gamble with three positive outcomes, each occurring with probability 1/3.*

is below the neutral weighting. The probability weighting goes from overestimation to underestimation for a value of p between 0.3 and 0.4.

CPT goes further and says that the weighting of a probability is also based on the rank of the corresponding outcome, *e.g.* whether the outcome is the best outcome, second best etc. With the CPT model, we associate more weight with the extreme outcomes. CPT considers the weighting of gains and losses separately. Thus, we divide $\mathbb{X}$ up into the subsets $\mathbb{X}^- = \{x_\perp^-, \ldots, x_\top^-\}$ and $\mathbb{X}^+ = \{x_\perp^+, \ldots, x_\top^+\}$ where $x_\top^- = x_\perp^+ = 0$ is the neutral outcome, $x_\perp^-$ is the worst loss possible and $x_\top^+$ is the best gain possible.

For gains, CPT's probability weighting is

$$\pi^+(x_\top^+) = w(p_\top^+) \tag{2.4}$$

$$\pi^+(x_i^+) = w(p_i^+ + \ldots + p_\top^+) - w(p_{i+1}^+ + \ldots + p_\top^+), \tag{2.5}$$

where $p_i = \Pr(x_i)$ is the probability of the outcome x_i occuring. For losses, the weighting is

$$\pi^-(x_\perp^-) = w(p_\perp^-) \tag{2.6}$$

$$\pi^-(x_i^-) = w(p_\perp^- + \ldots + p_i^+) - w(p_\perp^- + \ldots + p_{i-1}^-). \tag{2.7}$$

We can think of π as the marginal difference the outcome x_i makes to the function w. In the case of only positive outcomes, $\sum_x \pi^+(x) = 1$ [59].

To illustrate how weighting values are calculated, we consider the gamble shown in Figure 2.2. We have three positive outcomes, each occurring with probability 1/3. For this example, we use values for w based on experimental evidence [59]. According

to Equation 2.4, the weighting for $x_\top^+$ is

$$\begin{aligned}\pi^+(x_\top^+) &= w(1/3)\\ &\approx .367\end{aligned}$$

We next calculate the weighting values for outcomes $\pi^+(x_1^+)$ and $\pi^+(x_\perp^+)$, respectively, using Equation 2.5,

$$\begin{aligned}\pi^+(x_1^+) &= w(2/3) - w(1/3)\\ &\approx .56 - .367\\ &= .193,\end{aligned}$$

and

$$\begin{aligned}\pi^+(x_\perp^+) &= w(1) - w(2/3)\\ &\approx 1 - .56\\ &= .44.\end{aligned}$$

Despite the fact that each outcome occurs with equal probability, we show each outcome has a different weight associated with it. Specifically, we attach a higher weight to the extreme outcomes.

Based on these three ideas of loss aversion, framing and cumulative probability weighting, the overall value of the positive outcomes from a decision is

$$V(\mathbb{X}^+) = \sum_{x\in\mathbb{X}^+} \pi^+(x)u(x), \tag{2.8}$$

and the overall value of the negative outcome is

$$V(\mathbb{X}^-) = \sum_{x\in\mathbb{X}^-} \pi^-(x)u(x).$$

The overall value of a decision is then

$$V = V(\mathbb{X}^+) + V(\mathbb{X}^-).$$

Wakker *et. al.* provided an axiomatization of CPT without framing [66].[2] They showed that the key difference between EU and CPT was the independence axiom. CPT relies on *comonotonic independence*, a generalization of the independence axiom to a rank-dependent setting. This means that CPT is a strict generalization of EU.

[2]While the idea of framing has been used to understand many real world applications, we are unaware of any axiomatization of framing.

2.2 Preference Elicitation

In many areas of AI, we may be interested in making decisions on behalf of a user or recommending a decision to a user [22]. For example, we may want to create an optimal policy for a Markov Decision Process [52], help people make tough medical choices [20], or help people plan trips, taking into account the probabilities of delays [68].

In order to make or recommend intelligent decisions, we need to understand a user's preferences. We assume that a user's preferences are private; in general, the only source of information about a user's preferences is what they choose to tell us. (We can make additional assumptions but must be careful in doing so.) One possibility is to have a user state their utility values for each possible outcome. The cognitive burden of using this approach would be tremendous; the user would have to figure out the exact utility value for each outcome. A lot of this information might also be unnecessary. For example, for a specific set of decisions, we may be able to make an optimal decision knowing that $u(x_1) > 2u(x_2)$ but without knowing the exact value for $u(x_1)$ and $u(x_2)$.

The efficiency of eliciting preferences can be greatly increased if users only have to answer simple questions about their preferences. If a user's preferences follow EUT, a common technique is to use *standard gamble queries* (SGQs) [36].

Definition 9 (Standard Gamble Queries)**.** *A standard gamble query asks the user if they prefer the outcome x over the decision*

$$[1-p; x_\perp, p; x_\top].$$

We refer to this query as $q_x(p)$. Assuming normalized utility values between 0 and 1, if the person says they prefer the outcome over the decision, we can infer $u(x) > p$. Similarly, if the person says they prefer the decision over the outcome, we can infer $u(x) < p$. Otherwise, $u(x) = p$.

Example: An example of a real life standard gamble query occurs at the University of Waterloo used bookstore where students have two options when giving their books to the bookstore. They can either sell or consign the book. If a student sells the book, they receive an upfront amount of money. If the student consigns the book, they will receive a larger amount of money but only if the book is sold. For example, a student might be offered \$30 upfront or if they choose consignment, they could receive \$100 if the book is successfully sold and nothing otherwise. The student must estimate the probability of the book being sold. Ignoring issues related to temporal preferences, *i.e.* whether the student wants now or in the future, this is a standard gamble query.

For example, suppose the book store is offering to buy a used book for $30. If the book is succesfully sold on consignment the student will receive $80. The student believes there is a 70% chance that the book will be sold. To decide whether to sell the book or put it on consignment, the student must compare $u(\$30)$ against the gamble,

$$[0.3; u(\$0), 0.7; u(\$80)].$$

In making this comparison, the student is answering a SGQ. □

SGQs are simple yet powerful: to answer a SGQ, a person just has to decide either yes or no, yet we can use a series of SGQs to obtain an arbitrarily accurate bound on utility values.

Since SGQs will never give us the exact utility values (unless we have unbelievable luck in choosing p), we represent our limited knowledge about a user's utility value for the outcome x_i as a set of constraints:

$$[C_{\min}(x_i), C_{\max}(x_i)], \tag{2.9}$$

where $C_{\min}(x_i)$ is the minimum possible utility value for x_i and $C_{\max}(x_i)$ is the maximum possible utility. Initially, we set

$$[C_{\min}(x_\perp), C_{\max}(x_\perp)] = [0, 0],$$
$$[C_{\min}(x_\top), C_{\max}(x_\top)] = [1, 1],$$

and for all other $x_i \in \mathbb{X}$,

$$[C_{\min}(x_i), C_{\max}(x_i)] = [0, 1].$$

The complete set of constraints over $\mathbb{X}$ is $C \subseteq \mathbb{U}$.

2.2.1 Choosing the Best Decision

We next consider how to choose the optimal decision for a given set of decisions and utility constraints. "Optimal" can have several different definitions depending on our goals and what assumptions we make about the user.

In some cases, a user's preferences may be drawn from a known distribution of possible preferences. For example, we could be trying to help a pregnant woman decide whether to test her fetus for Down's syndrome [19, 20]. These tests carry a significant risk of miscarriage. There are several other factors that contribute to the value of the tests. Since most mothers have had to decide whether or not to have these tests, we can use their

preferences to create a probability distribution of possible preferences which can be used to help new mothers decide. In such cases, we let $\Pr(u)$ be the known distribution of possible utility values and $\Pr(u|C)$ be the distribution restricted to utility constraints C. If $\Pr(u)$ is known, the value of a decision is the *expected expected utility*, defined as [9]

$$EEU(d, \Pr(u|C)) = \sum_{x \in \mathbb{X}} \Pr_d(x) \mathcal{E}(x, \Pr(u|C)), \tag{2.10}$$

where $\mathcal{E}(x, \Pr)$ is the expected utility of the outcome x given the probability distribution $\Pr$. The rational choice is to then choose the decision which maximizes the expected expected utility [9]. The decision which maximizes EEU does not necessarily maximize EU. Similarly, the decision which maximizes EU does not necessarily maximize EEU.

An alternative criteria for choosing the optimal decision is *regret*. Regret is the expected loss of utility we experience from choosing one decision instead of an other, *i.e.*,

$$R(d, u) := \max_{d' \in \mathbb{D}} \left[EU(d', u) - EU(d, u) \right]. \tag{2.11}$$

Expected regret is the mean regret from choosing a decision given the probability distribution $\Pr(u|C)$. We define expected regret as

$$ER(d, \Pr(u|C)) = \int \left[EU(d^*_{EU}(u), u) - EU(d, u) \right] \Pr(u|C) du, \tag{2.12}$$

where $d^*_{EU}(u)$ is the decision which maximizes the EU given the utility values u.

Choosing the decision which minimizes the expected regret is another method for choosing an optimal decision. In fact, maximizing expected expected utility and minimizing expected regret are equivalent.

Lemma 2. *If decision d^* maximizes EEU, then d^* also minimizes ER.*

Proof. Proof by contradiction. We assume there exists a decision d^c such that $ER(d^c, \Pr(u|C) < ER(d^*, \Pr(u|C)$. By the definition of ER in Equation 2.12,

$$\begin{aligned} ER(d, \Pr(u|C))) &= \int \left[EU(d^*_{EU}(u), u) - EU(d, u) \right] \Pr(u|C) du \\ &= \int EU(d^*_{EU}(u), u) \Pr(u|C) du - \int EU(d, u) \Pr(u|C) du, \\ &= \int EU(d^*_{EU}(u), u) \Pr(u|C) du - EEU(d, \Pr(u|C)). \end{aligned}$$

Note that $\int EU(d^*_{EU}(u), u) \Pr(u|C) du$ is fixed for all decisions. Therefore, if $ER(d^c, \Pr(u|C) < ER(d^*, \Pr(u|C)$, this implies that $EEU(d^c, \Pr(u|C) > EEU(d^*, \Pr(u|C)$. □

Lemma 3. *If d^* minimizes ER, then d^* also maximizes EEU.*

Proof. Proof by contradiction. Assume there exists a decision d^C which has a higher expected expected utility than d^*. Then by the previous lemma, d^C also minimizes expected regret which is a contradiction. □

If we do not know $\Pr(u)$, we cannot calculate Equation 2.10, and therefore we cannot use expected expected utility or expected regret in helping to choose an optimal decision. If we do not know anything about the user's preferences other than the utility constraints C, then the reasonable option is to consider the worst-case scenarios from choosing each outcome.

For example, for a given set of constraints C, we could consider the worst case expected utility which is given by

$$EU_{\min}(d, C) = \min_{u \in C} EU(d, u).$$

In this case, we would then take the decision which maximized the worst-case expected utility.

When we are considering worst-case scenarios, it is more common to choose the decision which minimizes the worst-case regret. The best worst-case regret is known as *minimax regret* [10, 68]. To define worst case, or maximum, regret, we first define the *pairwise maximum regret* between decisions d and d' as

$$PMR(d, d', C) := \max_{u \in C} \left[EU(d', u) - EU(d, u) \right]. \tag{2.13}$$

The pairwise maximum regret is the most regret a user could experience from choosing decision d instead of d'. The maximum regret for decision d is then

$$MR(d, C) := \max_{d' \in D} PMR(d, d', C), \tag{2.14}$$

such that

$$r(d, u) \leq MR(d, C).$$

In this case, the minimax regret would be

$$MMR(C) := \min_{d \in D} \max_{d' \in D} PMR(d, d', C).$$

To achieve the minimax regret, we choose the decision

$$d^*(C) := \arg\min_{d \in D} \max_{d' \in D} PMR(d, d', C). \tag{2.15}$$

All we can say about the regret from choosing $d^*(C)$ is that it is at most $MMR(C)$. We do not know if $r(d^*(C), u)$ is equal to $MMR(C)$ or if $r(d^*(C), u)$ is considerably less.

Decreasing the minimax regret does not necessarily increase expected utility. For example, suppose we have two utility constraint sets C and C' such that $C' \subset C$. We know that $MMR(C) \geq MMR(C')$. However, it is not necessarily the case that $EU(d^*(C), u) \leq EU(d^*(C'), u)$. As our experimental results in subsequent chapters show, there are many times when $EU(d^*(C), u) > EU(d^*(C'), u)$. This is because minimax regret is focused on the worst case and expected utility is focused on the average case; improving the worst case does not necessarily improve the average case. However, $MMR(C)$ does provide a bound on how close our chosen decision is to maximizing expected utility.

Lemma 4. *For a given utility value u and utility constraint set C such that $u \in C$, let d^*_{MMR} be the minimax optimal decision and MMR be the resulting maximum regret. Let d^*_{EU} be the decision which maximizes expected utility. Then*

$$EU(d^*_{MMR}, u) \geq EU(d^*_{EU}, u) - MMR.$$

Proof. Proof by contradiction. Assume

$$EU(d^*_{MMR}, u) = EU(d^*_{EU}, u) - r,$$

where $r > MMR$. Then the pairwise regret from choosing d^*_{MMR} instead of d^*_{EU} is

$$EU(d^*_{EU}, u) - EU(d^*_{MMR}, u) = r.$$

The pairwise maximum regret from choosing d^*_{MMR} instead of d^*_{EU} is at least r. This means that the maximum regret from choosing d^*_{MMR} is at least r which is a contradiction. □

Therefore, if $MMR = 0$, we know that d^*_{MMR} also maximizes expected utility. However, if the expected regret is 0, this does not guarantee that d^*_{ER} maximizes expected utility. Therefore, if we want an absolute guarantee of the quality of a decision, we must use minimax regret.

2.2.2 Measuring the Confidence in Decisions

Given the uncertainty about a user's utility values, we can rarely guarantee that the chosen decision is actually the optimal decision. Instead, at best, we can say that the chosen decision is close to optimal. A key requirement in our work is that we offer a quantitative

measurement of "close:" we are not satisfied with statements such as "this seems like more or less the best decision." Both expected regret and minimax regret give a quantative measurement. With either measurement of regret, the resulting standard elicitation process is shown in Algorithm 1.

Algorithm 1 The standard preference elicitation algorithm. The actual regret can be either estimated or bounded by expected regret and minimax regret, respectively. The SGQ can be chosen at random or using the heuristics discussed in Section 2.2.3. We can either use EU maximization or maximum regret minimization as the metric for choosing the optimal decision as discussed in 2.2.1.

```
while regret given utility constraints C is greater than threshold do
    Query user using standard gamble query
    Use user's response to refine C
end while
Recommend optimal decision given C
```

2.2.3 Choosing the Best Query

If the expected or minimax regret is too high, we can ask the user additional queries to try to reduce the regret. Choosing the best query is a difficult process: the value of a query depends on the user's response, which we obviously do not know beforehand. A series of queries may also be more valuable than the sum of the values of each individual query [68]. As a result, we will rarely know the optimal query to ask; instead, we focus on heuristics to help us choose queries.

Our benchmark heuristic is the *halve largest-gap* (HLG) heuristic [10]. The HLG heuristic focuses on the outcome with the largest *utility gap* which is defined as

$$Gap(x) := C_{\max}(x) - C_{\min}(x). \tag{2.16}$$

If the outcome x^* has the largest utility gap, then we use the query $q_{x^*}((C_{\max}(x^*) + C_{\min}(x^*))/2)$. The HLG heuristic offers a guaranteed bound on the minimax regret after each query. This bound relies on a relationship between the minimax regret and the maximum utility gap of C which is defined as

$$maxGap(C) = \max_{x \in \mathbb{X}} \left[C_{\max}(x) - C_{\min}(x) \right]. \tag{2.17}$$

Proposition 1. *Given a constraint set C, the minimax regret is bounded by [10]*

$$MMR(C) \leq maxGap(C).$$

Boutilier *et. al.* originally proved this proposition for scenarios with non-risky decisions [10]. We use a novel approach to prove that the proposition still holds with risky decisions.

Proof. Proof by contradiction. In order for $MMR(C) > maxGap(C)$, $MR(d, C) > maxGap(C)$ must hold for all decisions. This means that for each decision d, there exists a decision d' such that $PMR(d, d', C) > maxGap(C)$. We bound $PMR(d, d', C)$ by

$$\begin{aligned} PMR(d, d', C) &= \max_{u \in C} \left[EU(d', u) - EU(d, u) \right] \\ &\leq \max_{u \in C} EU(d', u) - \min_{u \in C} EU(d, u) \end{aligned} \tag{2.18}$$

For brevity, let $EU_{\max}(d, C) := \max_{u \in C} EU(d, u)$ and let $EU_{\min}(d, C)$ be similarly defined.

Minimax regret can never increase as a result of refining the utility constraints. Therefore, we account for the maximum possible minimax regret by assuming that for all outcomes x_i, $Gap(x_i) = maxGap(C)$. We note that

$$\begin{aligned} EU_{\max}(d, C) &= \sum_{x \in \mathbb{X}} \Pr_d(x) C_{\max}(x), \\ &= \sum_{x \in \mathbb{X}} \Pr_d(x) \left[C_{\max}(x) - C_{\min}(x) \right] + \sum_{x \in \mathbb{X}} \Pr_d(x) C_{\min}(x), \\ &= \sum_{x \in \mathbb{X}} \Pr_d(x) \left[C_{\max}(x) - C_{\min}(x) \right] + EU_{\min}(d, C), \\ &= \sum_{x \in \mathbb{X}} \Pr_d(x) maxGap(C) + EU_{\min}(d, C), \\ &= maxGap(C) + EU_{\min}(d, C). \end{aligned} \tag{2.19}$$

We combine Equations 2.18 and 2.19 to show that

$$\begin{aligned} PMR(d, d', C) &\leq EU_{\max}(d', C) - EU_{\min}(d, C) \\ &\leq EU_{\max}(d', C) - EU_{\max}(d, C) + maxGap(C). \end{aligned}$$

In other words, to get $PMR(d, d', C) > maxGap(C)$, we need $EU_{\max}(d', C) > EU_{\max}(d, C)$. Similarly, since there exists some decision d'' such that $PMR(d', d'', C) > maxGap(C)$, we

know that $EU_{\max}(d'', C) > EU_{\max}(d', C)$. Repeating this process gives us a sequence of decisions, $\{d_1, d_2, \ldots\}$ such that $EU_{\max}(d_{i+1}) > EU_{\max}(d_i)$. Since the maximum expected utilities are strictly increasing, this sequence can never repeat a decision. This means the sequence must be infinite, which contradicts our model which has only a finite number of decisions. □

Proposition 2. *If the initial maximum utility gap is m, then with n outcomes, after $n\lceil\log(m/\epsilon)\rceil$ HLG queries, the minimax regret is at most ϵ.*

Our proof is a restatement of the proof given by Boutilier *et. al.* [10].

Proof. If we initially have $Gap(x_i) = m$, since each HLG query cuts the gap in half, after k HLG queries on x_i, we have $Gap(x_i) = 2^{-k}m$. If we want $Gap(x_i) \leq \epsilon$, then

$$\begin{aligned} 2^{-k}m &\leq \epsilon \\ \frac{m}{\epsilon} &\leq 2^k \\ \lg\left(\frac{m}{\epsilon}\right) &\leq k \end{aligned}$$

Therefore, we need $k = \lceil\log(m/\epsilon)\rceil$ to guarantee a utility gap of size ϵ. By Proposition 1, if we repeat this for all n outcomes, the resulting minimax regret will be at most ϵ. □

An alternative query selection heuristic is the *current selection* (CS) heuristic [10]. While the HLG heuristic is able to provide a guaranteed level of minimax regret, the CS heuristic is often able to outperform the HLG heuristic [10]. To describe the CS heuristic, we first define the *adversarial utility* as

$$u^a = \max_{u \in C} MR(d^*_{MMR}, u). \tag{2.20}$$

The adversarial utility is the utility that would result in the maximum actual regret (which would be equal to the minimax regret). We next define the *adversarial decision* as

$$d^a = \arg\max_{d \in \mathbb{D}} PMR(d^*_{MMR}, d, C). \tag{2.21}$$

If the user's utility values were equal to u^a, then d^a would be the decision that the user would have most wanted to choose over d^*_{MMR}. Since $PMR(d^*_{MMR}, d^a, C) = MMR(C)$, reducing $PMR(d^*_{MMR}, d^a, C)$ could be an efficient way of reducing $MMR(C)$. As a result, our new metric for choosing the best query is the *weighted utility gap*, defined as

$$|\Pr_{d^a}(x) - \Pr_{d^*}(x)|(C_{\max}(x) - C_{\min}(x)), \tag{2.22}$$

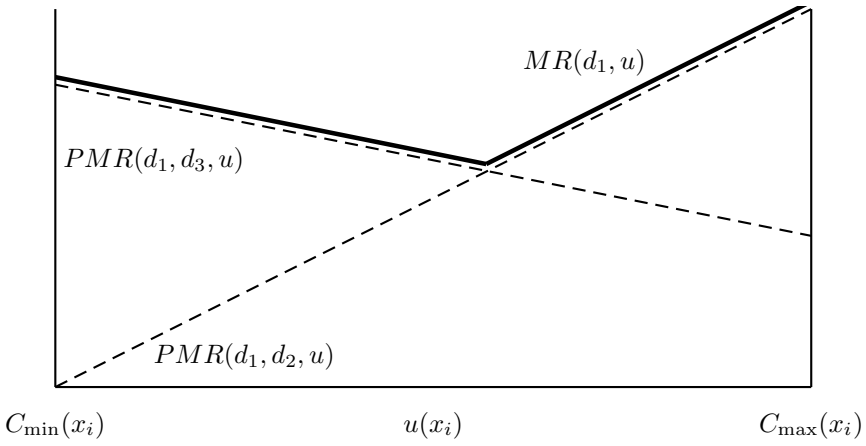


Figure 2.3: *An example of finding the maximum regret for decision d_1 as a function of $u(x_1)$ after the user's response to the query $q_{x_i}(p)$. In this example, $\mathbb{D} = \{d_1, d_2, d_3\}$. We first plot the PMR values for d_1 against both d_2 and d_3 as functions of $u(x_1)$ using Equation 2.23. The PMR values are shown as dashed lines. For example, since $PMR(d_1, d_2)$ is increasing, this means that $\Pr_{d_2}(x_i) > \Pr_{d_1}(x_i)$. The maximum regret, shown as the solid line, is the maximum of all the pairwise regret values.*

where $\Pr_{d^a}(x)$ is the probability of the outcome x occuring as a result of decision d^a and $\Pr_{d^*}(x)$ is similarly defined. If the outcome x^a maximizes the weighted utility gap, we use the query $q_{x^a}((C_{\max}(x^a) + C_{\min}(x^a))/2)$. To illustrate the advantage of the CS heuristic over the HLG heuristic, suppose for some outcome x, $\Pr_{d^a}(x) = \Pr_{d^*}(x)$. Then the size of the utility gap for x has no effect on $PMR(d^*, d^a, C)$ and therefore, reducing the utility gap will not reduce the minimax regret.

Both of the preceding heuristics query the user about the mid-point of the utility gap. A heuristic that can query the user about any utility value is the *maximin improvement* (MMI) heuristic [68]. Unlike the HLG and CS heuristics, MMI is not compatible with a monotonicity assumption. For every possible query value, MMI determines the improvement in the minimax regret if the user responds yes as well as the improvement if the user responds no. MMI then selects the query value which maximizes the minimum of the two improvements.

The MMI heuristic starts by expressing the PMR values as functions of $u(x_i)$, *i.e.* what would the PMR be if we knew $u(x_i)$ and for the rest of the outcomes, we only had the

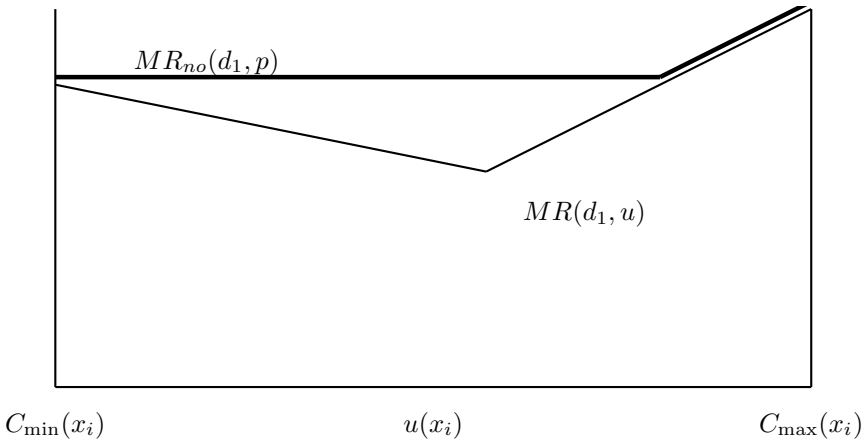


Figure 2.4: *Continuing the example from Figure 2.3. We consider the case where the user responds no to the query $q_{x_i}(p)$. Finding the resulting maximum regret for d_1 requires relaxing Equation 2.24 (shown as the thin line) to Equation 2.25 (shown as the thick line).*

utility constraints $C(\mathbb{X}\backslash x_i)$? In this case, the PMR would be

$$PMR(d, d', u(x_i)) = PMR(d, d', C(\mathbb{X}\backslash x_i)) + \left[\Pr_{d'}(x_i) - \Pr_{d}(x_i)\right] u(x_i), \tag{2.23}$$

with $C_{\min}(x_i) \leq u(x_i) \leq C_{\max}(x_i)$. For each decision d, we plot Equation 2.23 for every decision $d' \in \mathbb{D}\backslash\{d\}$. An example is shown in Figure 2.3. We can then express $MR(d)$ as a function of $u(x_i)$ by

$$MR(d, u(x_i)) = \max_{d' \in \mathbb{D}} PMR(d, d', u(x_i)). \tag{2.24}$$

An example of this function is also shown in Figure 2.3.

We now consider the case where the user responds no to the query $q_{x_i}(p)$, which implies that $u(x_i) < p$. If all we know is that $u(x_i) \in [C_{\min}(x_i), p)$ (instead of the exact value of $u(x_i)$), we must relax Equation 2.24 to be dependent on the range $[C_{\min}(x_i), p]$, *i.e.*

$$\begin{aligned} MR_{no}(d, p) &= \max_{d' \in \mathbb{D}} \max_{u \in [C_{\min}(x_i), p]} PMR(d, d', u) \\ &= \max_{u \in [C_{\min}(x_i), p]} MR(d_1, u). \end{aligned} \tag{2.25}$$

An example of relaxing Equation 2.24 to Equation 2.25 is shown in Figure 2.4. Therefore, we can calculate the maximum regret from choosing decision d_1 as a function of the utility value p, assuming the user responds no to the query $q_{x_i}(p)$.

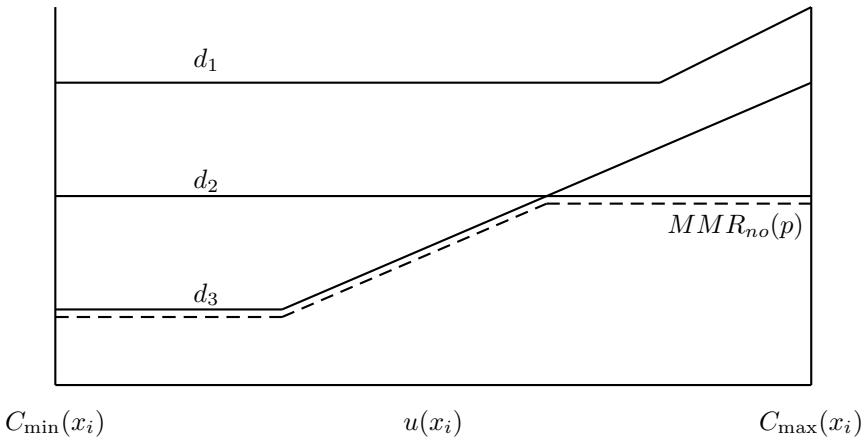


Figure 2.5: *Continuing our example from Figure 2.4, we next plot $MR_{no}(d, p)$ for each of the three decisions, i.e. the line d_2 represents $MR_{no}(d_2, p)$. We also take the minimum of $MR_{no}(d, p)$ at each value of p to get $MMR_{no}(p)$.*

We next plot Equation $MR_{no}(d, p)$ for each decision as shown in Figure 2.5. If we take the minimum of each of these values, we get $MMR_{no}(p)$, the resulting minimax regret if the user responds no to the query $q_{x_i}(p)$. The function $MMR_{no}(p)$ is shown in Figure 2.5 as the dashed piecewise line.

We can use analogous reasoning to find $MMR_{yes}(p)$. Figure 2.6 shows both MMR_{yes} and MMR_{no}. Since we do not know if the user is going to respond yes or no, we must assume the worst case and take the maximum of MMR_{yes} and MMR_{no}. Therefore, we focus on the *minimum improvement* which is defined as

$$MI(C, q_{x_i}(p)) = MMR(C) - \max\{MMR_{yes}(p), MMR_{no}(p)\}. \tag{2.26}$$

The value of p which maximizes the minimum improvement is the MMI query point. After finding the minimum improvement for querying each outcome, we select the query which maximizes Equation 2.26.

The MMI heuristic offers the best worst-case guarantees for individual queries, as opposed to HLG, which offers the best worst case guarantee for a long sequence of queries. However, there are times when the best worst-case is no improvement. In such cases, using MMI results in *stalling*, where MMI recommends a series of queries, each offering no improvement.

An alternative measure which is not subject to stalling is *expected improvement* (EI) [68].

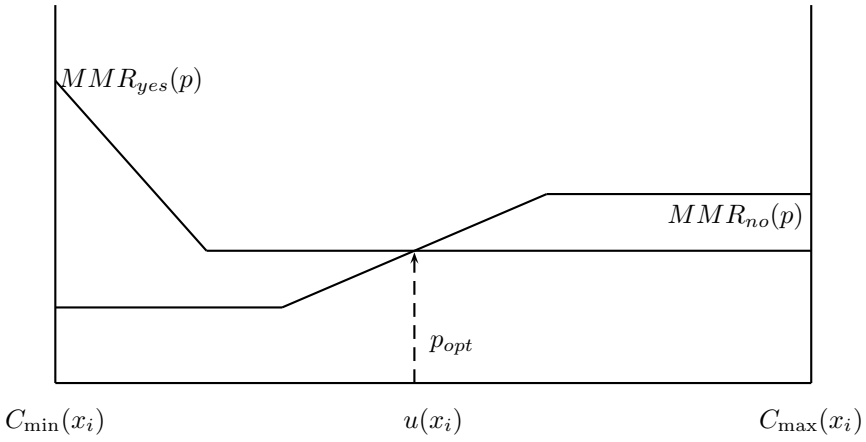


Figure 2.6: *Continuing the example from Figure 2.5. We plot both $MMR_{no}(p)$ and $MMR_{yes}(p)$. The intersection of these two functions gives the optimal worst-case querying value for the outcome x_i.*

The expected improvement from the query $q_{x_i}(p)$ given the utility constraints C is

$$\begin{aligned} EI(q_{x_i}(p), C) = MMR(C) &- \Pr(yes|q_{x_i}(p), C) MMR_{yes}(C, x_i, p) \\ &- \Pr(no|q_{x_i}(p), C) MMR_{no}(C, x_i, p). \end{aligned}$$

In this case, the *maximum expected improvement* (MEI) heuristic chooses the query which maximizes EI. We can accurately calculate $\Pr(yes)$ and $\Pr(no)$ if we have some estimation of the probability distribution of the user's preferences (in which case, we are in a setting similar to that assumed by expected regret). In this case, Monte Carlo sampling would be an efficient method for calculating $\Pr(yes)$ and $\Pr(no)$. If all we know about the user's preference is C, then we can still use MEI by assuming some probability distribution. Wang and Boutilier suggested assuming a uniform distribution, *i.e* [68]

$$\Pr(yes|q_{x_i}(p), C) = \frac{C_{\max}(x_i) - p}{C_{\max}(x_i) - C_{\min}(x_i)},$$

with $\Pr(no)$ similarly defined. The performance of the MEI heuristic assuming a uniform distribution will be heavily dependent on how closely the user's preferences are to being uniform.

2.3 Multiattribute Preferences

In many settings we may judge outcomes by different attributes. For example, if the possible outcomes are different cars, we may judge the outcomes by mileage, safety rating, number of doors, *etc.* Attributes can provide a more natural setting for people to reason about preferences: it can be difficult to compare two arbitrary cars, for example, since there are so many different ways cars can vary. Even if someone says they prefer car A over car B, this says nothing about whether they would prefer car C. If, however, the user says that each additional five kilometres per liter in fuel efficiency is worth $500 to them, this statement provides a powerful metric for comparing different cars.

Unfortunately, the number of possible outcomes grows exponentially with each additional attribute. For example, if we have n attributes and for each attribute, there are m possible outcomes, also known as *local outcomes*, we have m^n possible outcomes overall. As a result, there has been considerable research into compactly representing utility functions over multiple attributes [11, 25, 27, 36]. This work has focused on developing different models of utility independence.

Modelling utility independence has many similarities to modelling probability independence. In fact, some models of utility independence are directly inspired by Bayesian networks [11]. However, there is one key difference: Bayesian networks are considered the standard way of representing probability independence, while there is no standard in representing utility independence. Many models have been proposed which are fundamentally different from each other. In this section, we review some important models for multiattribute utility functions and compare their strengths and weaknesses.

2.3.1 Multiattribute Utility Functions

We begin by defining a model for a multiattribute setting. There exists some set of attributes $\mathbb{A} = \{A_1, \ldots, A_n\}$ that define all outcomes, *i.e.* $\mathbb{X} = A_1 \times \ldots \times A_n$. Without loss of generality, we assume that for each attribute there are m possible local outcomes. We make no assumption about the ordering of the outcomes for each attribute: thus, while a red sports car may be the best possible outcome, a green family car may be preferable to a red family car.

Models of Utility Independence

In this section, we review several *multiattribute utility independence* (MUI) models.

Additive Independence: The strictest form of independence is *additive independence* where

$$u(x) = \sum_{i=1}^{n} u_i(x) = \sum_{i=1}^{n} \lambda_i v_i(x),$$

where $u_i : A_i \to [0, 1]$ is a *subutility function* which is composed of a *scaling factor* $\lambda_i \in [0, 1]$ and a *local value function* (LVF) $v_i : A_i \to [0, 1]$ [36]. The key difference between a subutility function and a LVF is that there is always a local outcome x_i such that $v_i(x_i) = 1$. A key difference between additive independence and the other MUIs discussed below is that not all utility functions can be expressed using additive independence.

Example: Suppose we are flying from Toronto to Munich, Germany on Air Canada. There are several flights we can choose between. For example, we can fly from Toronto to Montreal and then to Munich or we can fly from Toronto to Frankfurt and then to Munich. We can use many different attributes to compare the utility of each flight chose. The most important attribute is probably cost. We might also be interested in the length of the total trip and whether or not our luggage gets lost during the trip. An example utility function based on these three attributes, assuming additive independence holds, is

$$u(x) = 0.8v_{cost}(x) + 0.15v_{length}(x) + 0.05v_{luggage}(x), \tag{2.27}$$

where x is a specific flight. For example, we could use Equation 2.27 to determine the utility of a trip with the attributes $[\$1500, 9hrs, lost]$, *i.e.* a trip which costs \$1,500, lasts 9 hours in total and results in lost luggage. □

Conditionally Additive Independence: A generalization of additive independence is *conditional additive independence* (CAI) [25]. A key idea in CAI is *features*.

Definition 10 (Features)**.** *Let $F^{att} \subseteq \mathbb{A}$ be a subset of all the attributes. A feature*

$$F = \times_{A_i \in F^{att}} A_i$$

is the set of all local outcomes over the set of attributes F^{att}. We can define subutility functions, scaling factors, and LVFs over features, that is, $u_F : F \to [0, 1]$, $\lambda_F \in [0, 1]$ and $v_F : F \to [0, 1]$. Note including attributes is an all or nothing selection, that is, we either include all local outcomes from an attribute A_i, or we do not include A_i. Since there is a direct mapping between F and F^{att}, we will use the terms interchangeably.

Suppose we have three independent features F_1, F_2, and F_3 such that $\mathbb{X} = F_1 \times F_2 \times F_3$. The features F_1 and F_3 are CAI given F_2 if there exist subutility functions $u_{F_1 \times F_2}(x)$ and

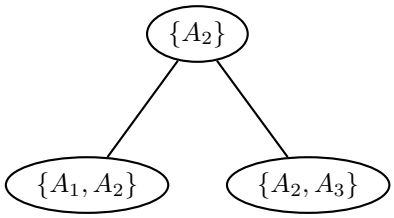


Figure 2.7: A CAI tree representation of the CAI factorization in Equation 2.29.

$u_{F_2\times F_3}(x)$ over features $F_1 \times F_2$ and $F_2 \times F_3$, respectively, such that

$$\begin{aligned} u(x) &= u_{F_1\times F_2}(x) + u_{F_2\times F_3}(x) \\ &= \lambda_{F_1\times F_2} v_{F_1\times F_2}(x) + \lambda_{F_2\times F_3} v_{F_2\times F_3}(x). \end{aligned} \tag{2.28}$$

We can apply CAI recursively to further decompose either $v_{F_1\times F_2}(x)$ or $v_{F_2\times F_3}(x)$. By definition $F_1^{att} \cap F_3^{att} = \emptyset$.

If we let $\mathbb{F} = \{F_1, \ldots, F_j\}$ be the set of all features in a CAI decomposition (or factorization), we can represent the decomposition graphically using a *CAI tree*. A CAI tree is a binary tree which shows the iterative factorization of the features. Each external node in a CAI tree corresponds to the attributes making up a feature in $\mathbb{F}$. Each internal node represents the intersection of its children nodes (empty intersections are allowed). The CAI tree does not have to be unique. For example, the CAI tree for the function

$$u(x) = u_{A_1\times A_2}(x) + u_{A_2\times A_3}(x) \tag{2.29}$$

is shown in Figure 2.7. We can use the CAI tree to determine F_1, F_2 and F_3 from Equation 2.28. Specifically, F_2^{att} is the root node, *e.g.* $F_2^{att} = \{A_2\}$, F_1^{att} is the first $\{A_1, A_2\}$ minus F_2^{att}, *e.g.* F_1^{att} is $\{A_1\}$ and by similar reasoning F_3^{att} is $\{A_3\}$.

Lemma 5. *Given a CAI decomposition with the set of features $\mathbb{F}$, for any resulting CAI tree, if we define $An(F_i, F_j)$ as the common ancestor of the features F_i and F_j, then $F_i^{att} \cap F_j^{att} = An(F_i, F_j)$.*

Proof. For this proof, we assume that $An(F_i, F_j)$ is the root of our CAI tree. It follows from definition that $An(F_i, F_j) \subseteq F_i^{att} \cap F_j^{att}$. We need to show that $F_i^{att} \cap F_j^{att} \subseteq An(F_i, F_j)$. We use proof by contradiction and assume that $F_i^{att} \cap F_j^{att} \supset An(F_i, F_j)$. We assume that F_i is in the left hand subtree rooted at $An(F_i, F_j)$ and F_j is in the right hand subtree. Let $\mathbb{F}_L^{att}$ be all of the attributes in our CAI tree which are in the left hand subtree rooted at

$An(F_i, F_j)$ and let $\mathbb{F}_R^{att}$ be all of the features in our CAI tree which are in the right hand subtree rooted at $An(F_i, F_j)$. We also define F_L^{att} as

$$F_L^{att} = \cup_{F \in \mathbb{F}_L^{att}} F$$

and F_R^{att} as

$$F_R^{att} = \cup_{F \in \mathbb{F}_R^{att}} F.$$

Since $F_i^{att} \subseteq F_L^{att}$ and $F_j^{att} \subseteq F_R^{att}$, we need only show that $(F_L^{att} \cap F_R^{att}) \supset An(F_i, F_j) = An(F_L^*, F_R^*)$ implies a contradiction. An alternative CAI factorization of u is

$$u(x) = u_{F_L}(x) + u_{F_R}(x). \tag{2.30}$$

In terms of Equation 2.28, Equation 2.30 breaks down into $F_2^{att} = F_L^{att} \cap F_R^{att}$, $F_1^{att} = F_L^{att} \backslash F_2^{att}$ and $F_3^{att} = F_R^{att} \backslash F_2^{att}$. By our assumption $F_1^{att} \cap F_3^{att} \neq \emptyset$ which contradicts our definition of CAI. □

Definition 11 (Cycle)**.** *A sequence of features $\{F_0, \ldots, F_n\}$ is a cycle if $F_i^{att} \cap F_{i+1}^{att} \neq \emptyset$ and $F_n^{att} \cap F_0^{att} \neq \emptyset$.*

Corollary 1. *Given a CAI decomposition, for any cycle, $F_i^{att} \cap F_j^{att} = F_s^{att} \cap F_t^{att}$ for any features F_i, F_j, F_s, F_t such that $F_i \neq F_j$ and $F_s \neq F_t$.*

In other words, if we have any cycles in a CAI decomposition, then the overlap between features in the cycle must be constant, that is, F_2 must be constant.

Example: Consider the decomposition

$$u(x) = u_{A_1 \times A_2}(x) + u_{A_1 \times A_3}(x) + u_{A_1 \times A_4}(x) + u_{A_1 \times A_5}(x). \tag{2.31}$$

Although there is a cycle in Equation 2.31 (in fact there are several), it is because the overlap between features is constant that Equation 2.31 is a valid CAI decomposition. The decomposition

$$u(x) = u_{A_1 \times A_2}(x) + u_{A_2 \times A_3}(x) + u_{A_3 \times A_1}(x), \tag{2.32}$$

is not a valid CAI decomposition since the overlap between features in the cycle is not constant. □

Generalized Additive Independence: We can further generalize additive independence with *generalized additive independence* (GAI) [27]. A set of features $\mathbb{F} = \{F_1, \ldots, F_m\}$ is generalized additive independent if we can express u as

$$u(x) = \sum_{F \in \mathbb{F}} u_F(x). \tag{2.33}$$

The key difference between CAI and GAI is that we place no restrictions on the features in $\mathbb{F}$ in Equation 2.33. Specifically, Lemma 5 does not apply to GAI. As a result, both Equations 2.31 and 2.32 are valid decompositions.

Given a GAI decomposition of u, we can construct a *GAI graph*. A GAI graph is an undirected $G_{GAI} = (\mathbb{F}, E)$ with edges between features F_i and F_j if and only if $F_i^{att} \cap F_j^{att} \neq \emptyset$ [12]. In other words, a GAI graph is a graph where each node is a feature in $\mathbb{F}$ and nodes are adjacent if and only if their corresponding features overlap.

Conditional Utility Independence: Given three independent features S, T and V such that $X = S \cup T \cup V$, S is *conditionally utility independent* (CUI) of T given V if for any fixed value of V, say v, [25]

$$\forall t \in T \; u(S, T, V) = u(S, T = t, V = v).$$

That is, for any fixed value of V, the user's preferences do not depend on T. As a result, we can decompose u into

$$u(S, T, V) = f(T, V) + g(T, V)u(S, T = t, V), \; g(\cdot) > 0. \tag{2.34}$$

Conditional Difference Independence: To define *conditional difference independence* (CDI_r), we first define a *reference utility function* [11].

Definition 12 (Reference Utility Function)**.** *Fix $x^r \in X$ to be a reference outcome (an obvious choice being $x^r = x_\perp$). Given a feature F, the reference utility function is*

$$u_F^r(x) = u(x_F \cup x_{\bar{F}}^r).$$

where x_F is the projection of x onto F and $\bar{F}^{att} = \mathbb{A} \backslash F^{att}$. That is $u_F^r(x)$, the utility where all attributes not in F are set to the reference outcome.

Given two features, F and F', a *conditional utility function* is

$$u^r(x_F | x_{F'}) = u_{F \cup F'}^r(x_F \cup x_{F'}) - u_{F'}^r(x_{F'}).$$

The feature F is *conditionally difference independent* (CDI_r) of F' given F'' if [11]

$$u^r(x_F | x_{F'}, x_{F''}) = u^r(x_F | x_{F''}).$$

This is denoted by $CDI_r(F, F' | F'')$. As a result, given any ordering of our set of attributes, we can define $Pa(A_i) \subseteq \{A_j | j < i\}$ to be the set of "parent" or conditional attributes such

that $CDI_r(A_i, \overline{Pa(A_i)}|Pa(A_i))$. As a result, we can express u as

$$\begin{aligned} u(x) &= u^r_{A_1}(x) + \sum_{i=2}^{n} u^r_{A_i}(x|Pa(A_i)) \\ &= \lambda_{A_1} v^r_{A_1}(x) + \sum_{i=2}^{n} \lambda_{Pa(A_i)} v^r_{A_i}(x|Pa(A_i)) \end{aligned}$$

An example of a CDI$_r$ representation over three attributes is shown in Figure BLAH.

Given a CDI$_r$ decomposition, we can construct a *CDI$_r$ graph*. A CDI$_r$ graph is a directed graph $G_{CDI_r} = (\mathbb{A}, E)$ with the edge $A_i \to A_j$ if and only if $A_i \in Pa(A_j)$ [11]. A CDI$_r$ graph has a strong relationship with Bayesian networks. Specifically, a directed CDI$_r$ graph captures the notion of *d-separation*.

2.3.2 Preference Elicitation with MultiAttribute Preferences

Since MUI models allow for compact representations of multiattribute preferences, we would like to have preference elicitation processes which take advantage of the MUI models. While we could always use basic SGQs by converting a MUI decomposition into a standard model (as discussed in Section 2.1.1), this would defeat the purpose of using a MUI model. For GAI, and by extension, additive independence and CAI, we can adopt SGQs to take advantage of the structure. For both models, queries are needed to bound both the scaling factors and the LVFs.

For the feature F, $\lambda_F = u_F^{\max} - u_F^{\min}$ where $u_F^{\max} = max_{x_F \in F} u_F(x_F)$ and $u_F^{\min}$ is similarly defined [13]. Therefore, to bound λ_F, we need to bound both $u_F^{\max}$ and $u_F^{\min}$. To bound these two values, we use traditional SGQs. When eliciting multiattribute preferences, we refer to the basic SGQs as *global SGQs* [13]. To bound $v_F(x_F)$ for local outcomes x_F, we use *local standard gamble queries* (lSGQs). A lSGQ asks the user if they would prefer the decision $[1-p; x_F^{\min}, p; x_F^{\max}]$ over the local outcome x_F, assuming all attributes not in F are fixed. If the user does prefer the guaranteed outcome, then $v_F(x_F) > p$ [13].

Example: We have a user flying from Toronto to Munich, Germany. We are trying to elicit the user's utilities to help them choose the best flight. This user judges flights on three attributes; cost, length and whether or not the luggage is lost. Suppose the user's overall utility function based on these three attributes follows additive independence and is given by Equation 2.27.

We wish to query the user about their utility with regards to the length of the flight. In the best case, the trip will take 8 hours and the in the worst case the trip will take 24 hours. (Although the trip could actually take longer if, for example, a volcano erupted and cancelled all flights for 4 days, we decide that the probability of such events occuring is low enough that we can ignore them.) For this example, we wish to query the user about their utility of a 10 hour trip. To do so, we would give the user a lSGQ comparing a trip with a guaranteed length of 10 hours against a trip with a length given by the gamble

$$[1-p; 24hrs, p; 8hrs],$$

i.e. with probability 1-p, the trip would take 24 hours and otherwise would take 8 hours. This query would assume that the cost of the trip and whether or not the luggage was lost is fixed. □

Previous Comparisons of Utility Independence

The compactness of a utility decomposition is the size of the decomposition. For example, a decomposition could have a compactness of $O(m^3)$ using a GAI representation, that is, if we use GAI, then the size of the resulting decomposition is $O(m^3)$. To the best of our knowledge there has been little work on comparing the compactness of different MUI models.

Additive independence is the strongest form of multiattribute independence, and therefore, the least compact. All of the independence models discussed in this section generalize on additive independence. Engel and Wellman showed that CUI is a weakly more compact representation than CAI, *i.e.* CUI is always at least as compact as CAI and sometimes more so [25]. However, they did not examine cases where GAI achieves a more compact representation than CAI. They dismissed such cases as "hard to identify" and lacking "an intuitive interpretation" [25].

Chapter 3

Preference Elicitation and Cumulative Prospect Theory

Preference elicitation in AI traditionally assumes the user follows expected utility. In Section 3.1, we examine why expected utility is assumed. Since cumulative prospect theory is a better descriptive model of peoples' actual preferences we also investigate whether current AI preference elicitation techniques are compatible with cumulative prospect theory; we show that they are in fact incompatible. In Section 3.3, we introduce a new method for preference elicitation that is compatible with cumulative prospect theory. Experimental results are given in Section 3.5.

3.1 The Role of Preference Models in Preference Elicitation

Papers in AI which deal with preference elicitation over risky outcomes make two strong assumptions about *expected utility* (EU). The first assumption is that EU is a valid normative or rational model, *i.e.* EU is a model of how people should reason about preferences. The second assumption is that EU is a valid descriptive model, *i.e.* EU is a model of how people actually reason about preferences.[1]

[1]There is an important difference between EU and expected utility theory (EUT). EUT says that EU holds if both decomposability and the independence axiom hold. As shown in Section 2.1.1, EUT is always correct and does not have to be assumed. If we assume EU holds, then we are assuming that decomposability and the independence axiom hold.

These assumptions are made for two different reasons. Assuming EU is a valid rational model means that maximizing EU is a rational goal. This is important because preference elicitation papers either aim to maximize EU or optimize some metric closely related to EU [20, 68]. People have generally agreed that EU is a valid rational model and therefore, this assumption is uncontentious [54, 55].

Assuming EU is a valid descriptive model provides a framework for how to interact with people. Specifically, by assuming EU, we can interpret a person's response to the *standard gamble query* (SGQ). Unfortunately, as discussed in Section 2.1.1, EU is a poor descriptive model. This has lead to numerous alternative models which attempt to be accurate descriptive models. Out of all of these alternative descriptive models, *cumulative prospect theory* (CPT) is considered the best model [55].

Our goal in this chapter is to develop a preference elicitation method which always works with users whose preferences follow CPT. Since there is general agreement that EU is a valid normative model, we will keep EU as our normative model. As a result, while we assume that user's preferences follow CPT, we believe that users should actually be trying to maximize their expected utility.

We begin by discussing the relationship between EU and SGQS. If a person's preferences followed EU, then they would evaluate the overall utility of the SGQ $q_x(p)$ as

$$\begin{aligned} U([1-p; x_\perp, p; x_\top]) &= EU([1-p; x_\perp, p; x_\top]) \\ &= (1-p)u(x_\perp) + pu(x_\top), \\ &= p. \end{aligned}$$

Therefore, if the user says that they prefer the certain outcome x over the given gamble, we can infer that $u(x) > p$.

Without the assumption of EU, the relationship between $u(x)$ and the user's response to the SGQ is no longer as clear. For example, if we assume that the user preferences follow only CPT, then assuming that $x_\perp > 0$, from Equation 2.8 the user's evaluation of a given gamble is now

$$\begin{aligned} U([1-p; x_\perp, p; x_\top]) &= (w(1) - w(p))u(x_\perp) + w(p)u(x_\top), \\ &= w(p). \end{aligned} \tag{3.1}$$

Without knowing how the user weights probabilities, we can no longer infer any relationship between the $u(x)$ and p.

The easiest solution is to ignore the probability weighting, *i.e.* assume $w(p) = p$. This means that while the user is weighting the probabilities, we are analyzing their responses

as if they are not. To see if this approach could work, we consider a scenario with four outcomes $\mathbb{X} = \{x_\perp, x_1, x_2, x_\top\}$ and two decisions, d and d';

$$[1-p; x_1, p; x_\top], \tag{3.2}$$

and

$$[1-p'; x_\perp, p'; x_2], \tag{3.3}$$

respectively.

Our goal is to choose a set of utility values and probabilities so that there is an error with the minimax regret. If we can create an error, this would show that we cannot ignore the probability weighting. Since minimax regret is supposed to be an upper bound on the actual regret, there is an error any time the actual regret is greater than the minimax. Our measure of error is

$$MMR_{err} = \max\{0, \text{actual regret} - \text{minimax regret}\} \tag{3.4}$$

Hence, our goal is to choose utility values and probabilities that result in Equation 3.4 being greater than zero.

Suppose we query the user until we believe we know the user's exact utility values for both x_1 and x_2. In this case, we believe that the utilities for outcomes x_1 and x_2 are $w(u(x_1))$ and $w(u(x_2))$, respectively. For brevity, we will refer to these values as w_1 and w_2. If we believe we know the exact utility for all possible outcomes, the minimax regret will be zero. Therefore, an error occurs if we can get the actual regret to be positive. For our example, we will consider the situation where the user prefers d' but the minimax decision is d. For the reverse situation, the reasoning is analogous.

According to the incorrect utility values w_1 and w_2, we incorrectly believe the regret from choosing decision d is

$$\begin{aligned} R(d, u_{incorrect}) &= r(d, d', u_{incorrect}) \\ &= EU(d', u_{incorrect}) - EU(d, u_{incorrect}) \\ &= p'w_2 - (p + (1-p)w_1), \end{aligned} \tag{3.5}$$

where $u_{incorrect} = [u_{incorrect}(x_1) = w_1, u_{incorrect}(x_2) = w_2]$. In order for d to be the minimax decision, Equation 3.5 must be less than or equal to 0. This can always be achieved by setting $w_2 = w_1 = 0$. Therefore, there always exist values for w_1 and w_2 such that d appears to be the preferred decision, even if d' would actually maximize the user's expected utility.

The actual regret from choosing d is

$$p'u(x_2) - (p + (1-p)u(x_1)). \tag{3.6}$$

Since $MMR = 0$, for MMR_{err} to be greater than zero, Equation 3.6 just needs to be greater than zero. We maximize Equation 3.6 by setting $u(x_2) = 1$ and $u(x_1) = 0$ which reduces the equation to

$$p' - p. \tag{3.7}$$

Therefore, anytime that

$$\begin{aligned} p' - p &> 0 \\ p' &> p, \end{aligned} \tag{3.8}$$

the user prefers decision d' over d, resulting in $MMR_{err} > 0$. Since $MMR_{err} > 0$, minimax regret is no longer a valid upper bound on the actual regret.

We can get an arbitrarily large error, *i.e.* $MMR_{err} = 1$, by setting $p = 0$ and $p' = 1$. If Equation 3.8 is not satisfied, we can still create an error by choosing values such that Equation 3.5 is greater than 0 and Equation 3.6 is less than 0. Thus in theory, for any given set of gambles of the form of Equation 3.2 and 3.3, there always exist utility values and probability weighting values which result in an error if we ignore the probability weighting. This shows that if the user is applying probability weightings, we cannot ignore them.

However, these errors may not be realistic since setting $w_1 = 0$ is not realistic. In order to examine more realistic settings, we need to use probability weighting functions built on experimental experience. We examined these realistic scenarios in the following experiments.

We begin by using the weighting function [59]

$$w(p) = \frac{p^\gamma}{(p^\gamma + (1-p)^\gamma)^{1/\gamma}}. \tag{3.9}$$

Compared to other weighting functions, Equation 3.9 has been shown to be relatively accurate at modelling peoples' weightings [71]. We used a value for γ of 0.71 with a standard deviation of 0.1 [71].

With these realistic values for w, we then searched for possible scenarios where the minimax regret was violated, *i.e.*, Equation 3.4 was greater than zero. This search was based on examining possible sets of decisions in the form given by Equations 3.2 and 3.3. To do so, we consider values of p and p' both ranging from 0.1 to 0.9, inclusive, in

Number of Standard Deviations	Percent Error	Max Error Tversky-Kahneman w	Max Error Prelec w
1	44.4	0.215	0.155
2	44.4	0.315	0.245
3	44.4	0.444	0.35

Table 3.1: *Error rates for preference elicitation using SGQs on users with CPT-modelled preferences. Percent error is the percentage of decisions in the form given by Equation 3.2 and 3.3, for which there exist utility and probability weighting values which result in the minimax regret being violated. Error is calculated according to Equation 3.4. The probability weighing was calculated using the forms for w given by Tversky and Kahneman (Equation 3.9) and Prelec (Equation 3.10). Results are given for different ranges of γ, given in terms of the number of standard deviations around the mean. For example, in the first row, for Tversky and Kahneman's w, γ ranges from 0.61 to 0.81 while for Prelec's w, γ ranges from 0.60 to 0.88.*

increments of 0.1. For each set of decisions, we searched for any utility and probability weighting values that resulted in Equation 3.5 being less than 0 and Equation 3.6 being greater than 0 (or vice-versa). We searched through all possible utility values ranging from 0.05 to 0.95 in increments of 0.05. For probability weighting values, we searched using values of γ in increments of 0.01 between lower and upper bounds given in terms of the standard deviation of γ. For example, if we set the upper and lower bounds of γ to be plus or minus one standard deviation, then we would examine values of γ from 0.61 to 0.81. (The same γ would be used in calculating both w_1 and w_2.) If any such scenarios exist, we measure the error according to Equation 3.4.

The results of this search are shown in Table 3.1. We see that, regardless of the range of possible values for γ, for 44% of all decisions, there exists at least one set of utility and probability weighting values which result in the violation of the minimax regret. For values of γ within one standard deviation, *i.e.* values of γ which cover 68.2% of people, the maximum error is 0.215. For values of γ within three standard deviations, *i.e* values of γ which cover 99.8% of people, the maximum error is 0.444. These values are significant and show that the assumption of expected utility has important consequences. The fact that the percent error is unaffected by the range of γ suggests that, in practice, only certain decisions can result in the violation of minimax regret. This suggests that certain decisions are "immune" to the effects of probability weighting.

Since there is some disagreement about the form of the probability weighting function,

Number of Standard Deviations	Percent Error	Max Error Tversky-Kahneman w	Max Error Prelec w
1	100	0.215	0.155
2	100	0.315	0.245
3	100	0.444	0.35

Table 3.2: *A repeat of the results from Table 3.1 with a monotonicity constraint imposed on u and the requirement that $p' > p$.*

we next tested a different weighting function,

$$w(p) = e^{-(-\lg p)^{\gamma}} \tag{3.10}$$

proposed by Prelec [49]. Experimental results have suggested that while, on average, Prelec's w may not be as accurate as Tversky and Kahneman's, there are cases where Prelec's is more accurate [71]. Based on experimental results, the mean and standard deviation for γ with respect to Equation 3.10 are 0.74 and 0.14, respectively [71]. The results are also shown in Table 3.1. The percentage of decisions with which an error can occur is identical to the percentage found using Tversky and Kahneman's w. This implies that whether or not minimax regret can be violated is not dependent on the exact form of the probability weighting. However, the maximum error is slightly lower with Prelec's w.

We wanted to see how robust these results are. Since a monotonicity constraint is often a natural constraint, we next added a monotonicity constraint that $u(x_1) < u(x_2)$. We also added the constraint that $p' > p$. (With the monotonicity constraints, if $p' \leq p$, the user will always prefer the decision in Equation 3.2 and so we would never bother processing a user for such a scenario.) While this setup guarantees that Equation 3.6 is always greater than 0, it does not guarantee that Equation 3.5 will be less than 0. Thus while an error is not guaranteed, the results, shown in Table 3.2, show that in practice, it is always possible to have an error. The monotonicity constraints increase the probability of an error occurring but do not affect the magnitude of the error. As a result, in scenarios with monotonicity constraints, assuming EU can always create an error.

3.2 Previous Methods of Preference Elicitation with Cumulative Prospect Theory

Over the last 25 years, there has been considerable work on trying to elicit utilities when expected utility may not hold [26]. This has lead to several elicitation methods which are

specifically compatible with CPT. In this section we review the methods and discuss their advantages and disadvantages.

The first possible method is to learn and model the user's probability weightings. If we knew the user's weighting function, we could continue using SGQs. One method for finding the user's weighting function is a parametric approach where we assume that the functional form of w is known. In this case, we can query the user directly about their probability weighting [71]. Unfortunately, to our knowledge, no functional form of w has been proposed which is able to explain experimental results to any accurate degree. For example, while the probability weighting function in Equation 3.9 has been used to study peoples' probability weighting in many different experiments, studies have found that this function is able to explain only 39% of the variation from a neutral probability weighting [71]. While there are weighting functions that are able, on average, to explain more of the variation, these functions are not consistently better. For example, the weighting function proposed by Kahneman and Tversky in their *original prospect theory* (OPT) paper has been shown to explain around 47% of the variation [71]. However, Wu and Gonzalez showed that there are probability values for which CPT provides a better explanation than OPT [71].

An alternative method for modelling the weighting function is a non-parametric approach where we do not assume any functional form for w. There are two possible non-parametric approaches to eliciting probability weightings. The advantage of both approaches is that there is no error created by trying to fit the user's responses to a functional form of w.

The first non-parametric approach was proposed by Bleichrodt and Pinto [7]. We note that the disadvantage of their approach is that they assume the utility values have already been elicited. As a result, if we use Bleichrodt and Pinto's method to find the user's weighting values which we then use in SGQs, we would be eliciting the user's preferences twice, creating unnecessary work.

Bleichrodt and Pinto's method uses the the *gamble-tradeoff method* (GT method) to find the user's preferences [7, 65]. The GT method is able to elicit users' preferences even when users distort probabilities in their utility evaluations. This means that we could use the GT method by itself, instead of SGQs, to elicit the preferences. The GT method assumes that $\mathbb{X}$ is a subset of a larger set of continuous outcomes $\mathbb{Y}$ which has no upper bound. For example, $\mathbb{X}$ could be different amounts of money to be won with the amount determined by the roll of a die, *e.g.* if the die rolls 1, we win \$1 etc. In this case, $\mathbb{X} = \{\$1, \ldots, \$6\}$ and we could have $\mathbb{Y} = \{\$y | y \in \mathbb{R}_{\geq 0}\}$. As a result, the user has a utility for winning \$4.5 or \$10, even though neither outcome is possible for this specific scenario. This means that we can generalize u to be over the domain $\mathbb{Y}$. The GT method also assumes that u is

monotonically increasing.

The GT method starts by asking the user to choose an outcome $y_1 \in \mathbb{Y}$ such that they are indifferent between the decisions

$$[p; x_\perp, 1-p; R] \text{ and } [p; y_1, 1-p; r], \tag{3.11}$$

where the outcomes $r \prec R$ are fixed reference outcomes in $\mathbb{Y}$ and the probability p is also fixed. We then ask the user to choose an outcome $y_2 \in \mathbb{Y}$ such that they are indifferent between the decisions

$$[p; y_1, 1-p; R] \text{ and } [p; y_2, 1-p; r], \tag{3.12}$$

where r, R and p are the same values used in the previous comparison. We evaluate both indifferences according to CPT. Equation 3.11 evaluates to

$$\begin{aligned} w(1-p)u(R) + (1-w(1-p))u(x_\perp) &= w(1-p)u(r) + (1-w(1-p))u(y_1), \\ w(1-p)u(R) - w(1-p)u(r) &= (1-w(1-p))(u(y_1) - u(x_\perp)), \end{aligned} \tag{3.13}$$

and Equation 3.12 evaluates o

$$\begin{aligned} w(1-p)u(R) + (1-w(1-p))u(y_1) &= w(1-p)u(r) + (1-w(1-p))u(y_2), \\ w(1-p)u(R) - w(1-p)u(r) &= (1-w(1-p))(u(y_2) - u(y_1)). \end{aligned} \tag{3.14}$$

Together, Equations 3.13 and 3.14 imply that

$$u(y_1) - u(x_\perp) = u(y_2) - u(y_1),$$

which means that $u(y_1)$ is halfway in between $u(x_\perp)$ and $u(y_2)$. We can repeat this process to give us $\{y_1, y_2, \ldots, y_l\}$. We normalize the utility values such that $u(y_l) = 1$. Since the utility values for y_j are evenly spaced, this means that $u(y_j) = j/l$. We then set the utility constraints for $u(x_i)$ to be

$$[u(y_{j-1}), u(y_j)] \text{ such that } y_{j-1} \preceq x_i \prec y_j. \tag{3.15}$$

We control the accuracy of these constraints by changing the values for the parameters p, r and R in order to bring the outcomes y_i and y_{i+1} closer together. While choosing an outcome which results in indifference between decisions (as done in Equations 3.11 and 3.12) takes more cognitive effort than responding yes/no to a SGQ, the GT method has been used successfully in several experiments with people [2, 5, 65]. This shows that the GT method is a practical method for eliciting preferences.

However, the GT method is not efficient at reducing expected or minimax regret. The accuracy of the results is controlled by how close the outcomes $\{y_1, y_2, \ldots, y_l\}$ are together. The closer these outcomes are together, the more accurate the utility constraints for outcomes in $\mathbb{X}$ are. The first problem is that the GT method does not provide utility constraints for all outcomes until $y_l \succ x_\top$. For example, suppose $\mathbb{X} = \{0, 25, 75, 100\}$ and $y_l = 50$. In this case, we cannot use Equation 3.15 to bound $u(75)$ or $u(100)$. All we can say is that $u(75) > 1$ and $u(100) > 1$ which are not useful bounds. This means that we cannot calculate expected or minimax regret until we have completed the GT method. At this point, if the regret is too high, we must restart the GT method over from the beginning. Since the accuracy of the constraints is controlled by three independent parameters, it is not clear what are the best parameter values to choose to improve accuracy. Once we have changed any of the three parameters, our y-values will not match up with the y-values from the previous run. Therefore, every time we restart the GT method, we must discard our previous set of constraints. Many of the outcomes in $\{y_1, y_2, \ldots, y_l\}$ are not of use to us. For example, suppose $\mathbb{X} = \{0, 75, 100\}$ and for our given scenario, to achieve the desired minimax regret, the maximum allowable utility gap around $u(75)$ corresponds to $u(74)$ and $u(76)$. In this case, we may need $\{y_1 = 1, y_2 = 2, \ldots, y_{74} = 74, y_{76} = 76, \ldots\}$. This would result in a lot of unnecessary querying. Therefore, using the GT method to minimize regret can be both inefficient and unintuitive to optimize.

The second non-parametric approach for eliciting probability weightings, the *midweight method*, is by van de Kuilen and Wakker [61]. The midweight method is based on the GT method. The advantage of the midweight method is that we only need to find one pair of outcomes y_1 and y_2 such that $u(y_2) = 2u(y_1)$. This means that we only need to perform one iteration of the GT method. Once these two outcomes have been found, the midweight method asks users to find a probability d such that they are indifferent between the decisions

$$[b; x_\perp, c; y_1, a; y_2] \text{ and } [b + (c - d); x_0, a + d; y_2],$$

for probabilities a, b and c such that $a + b + c = 1$ and $0 < d < c$. Van de Kuilen and Wakker showed that [61]

$$w(d + a) = \frac{w(a) + w(c + a)}{2}.$$

For example, if we choose $a = 0$ and $c = 1$ then since $w(a) = 0$ and $w(a + c) = 1$, we can use the midweight method to find $d = w^{-1}(0.5)$. While van de Kuilen and Wakker did not use the midweight method to minimize regret, it is straightforward to combine the midweight method and SGQs. Suppose we give the user the SGQ $q_{x_i}(p)$ and they respond "no" (they prefer the decision over x_i). As shown in Equation 3.1, we can only infer that $u(x_i) < w(p)$. Since w is monotonically increasing, we can use the midweight method to

bound $w(p)$. For example, if $p < w^{-1}(0.5)$ then $w(p) < 0.5$. Each time we query the user, we can either update our utility constraints by using a SGQ or we can update our probability weight bounds by using a midweight query.

One possibility with the midweight method is to learn the user's probability weighting in a simpler setting. However, peoples' probability weighting will vary from situation to situation [65]. In one test, peoples' probability weighting was elicited for situations involving money and for situations involving medical options resulting in different life expectancies [65]. People were shown to apply heavier probability weightings to the situations involving medical options. This means that we cannot necessarily learn peoples' probability weightings in one situation and apply them to another.

3.3 The Gamble Equivalence Method

We propose the *gamble equivalence method* (GEM), a preference elicitation method based on the gamble-tradeoff method, which is compatible with CPT and which focuses on reducing the minimax regret as quickly as possible. GEM uses two new types of queries: *configuration queries* and *outcome queries*.

Configuration queries: These queries are used to find out enough about the user's probability weighting so that we can properly interpret the results of the outcome queries.

Outcome queries: Once we have enough information about the user's probability weighting, we can use outcome queries to query the user about their utility values.

3.3.1 The Scenario

The scenario for using GEM is based on the user model presented in Section 2.1. A user faces a set of possible outcomes $\mathbb{X} = [x_\perp, \ldots, x_\top]$. The user has a private utility function u such that $u(x_\perp) = 0$ and $u(x_\top) = 1$. As with the GT method, we assume that the set $\mathbb{X}$ is a subset of the continuous set $\mathbb{Y}$ which has no upper bound. The user has a utility value for all outcomes in $\mathbb{Y}$, *i.e.* we can generalize u to be $u : \mathbb{Y} \to \mathbb{R}$. We assume that u is monotonically increasing and $\lim_{y\to\infty} u(y) = \infty$. These are common, though not universal, assumptions in many economic scenarios [41].

The user evaluates their preferences according to CPT. The user's weighting function w is also private though we assume that w is monotonically increasing, an assumption

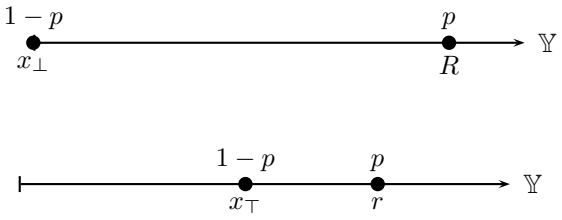


Figure 3.1: *A graphical representation of the query in Equation 3.18. The user is asked to provide a probability p such that they are indifferent between these two decisions.*

supported by experimental evidence [71]. We assume that EUT is a valid normative model. Since we do not assume that the user's preferences are drawn from any known distribution, our goal is to minimize minimax regret.

3.3.2 Configuration Queries

The goal of configuration queries is to find out the necessary information about the user's probability weighting so that we correctly interpret the user's response to our outcome queries by factoring out the effects of probability weighting. We factor out the effects of probability weighting by solving the equation

$$\frac{w(p^*)}{w(1-p^*)} = \frac{1}{2}, \tag{3.16}$$

for the probability p^*. The reason why we only need to solve Equation 3.16 is algebraic. We use p^* in our outcome queries (shown in the next section). With the equations that result from the user's evaluation of outcome queries, we are able to isolate the probability weighting terms in the factor

$$\frac{w(p^*)}{w(1-p^*)}. \tag{3.17}$$

By solving Equation 3.16, we are able to replace Equation 3.17 in the evaluation of the outcome queries with 1/2, which removes the effects of w. It is possible to solve Equation 3.16 for values other than 1/2. Using 1/2 gives the best guaranteed improvement; *i.e.* we know that regardless of the user's response to an outcome query, we will be reducing a utility gap by half.

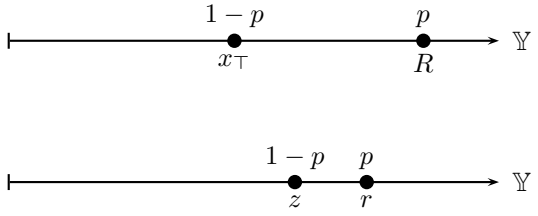


Figure 3.2: *A graphical representation of the query in Equation 3.19. The user is asked to provide an outcome z such that they are indifferent between these two decisions.*

The first step in solving Equation 3.16 is to find an outcome $z \in \mathbb{Y}$ such that $u(z) = 2$. Since we assumed that $\lim_{y\to\infty} u(y) = \infty$, we know z exists. To find z, we use a method based heavily on the GT-method from Section 3.2. We start by picking two outcomes r and R in $\mathbb{Y}$ such that $x_\top \prec r \prec R$. We then ask the user to pick a probability p' such that they are indifferent between the decisions

$$[1-p'; x_\bot, p'; R] \text{ and } [1-p'; x_\top, p'; r]. \tag{3.18}$$

This can either be done by asking the user directly for p' or using a series of queries similar to SGQs. A graphical representation of this query is shown in Figure 3.1. We next ask the user for an outcome $z \in \mathbb{Y}$ such that they are indifferent between the decisions

$$[1-p'; x_\top, p', R] \text{ and } [1-p'; z, p', r]. \tag{3.19}$$

Due to the rank-dependent nature of CPT, we require that $x_\top \prec z \prec r$. If this is not the case, we must increase r and R and repeat these two steps. A graphical representation of this query is shown in Figure 3.2.

It follows from the indifference between the decisions in Equation 3.18 that [2]

$$\begin{aligned}[w(1)-w(p')]u(x_\bot) + w(p')u(R) &= [w(1)-w(p')]u(x_\top) + w(p')u(r),\\ \Rightarrow w(p')[u(R)-u(r)] &= [1-w(p')][u(x_\top - u(x_\bot)].\end{aligned} \tag{3.20}$$

Similarly, it follows from the indifference between the decisions in Equation 3.19 that

$$\begin{aligned}[w(1)-w(p')]u(x_\top) + w(p')u(R) &= [w(1)-w(p')]u(z) + w(p')u(r),\\ \Rightarrow w(p')[u(R)-u(r)] &= [1-w(p')][u(z)-u(x_\top)].\end{aligned} \tag{3.21}$$

[2]This derivation closely follows the derivation used with the GT-method shown in Section 3.2.

Combining Equations 3.20 and 3.21, we get

$$u(z) - u(x_\top) = u(x_\top) - u(x_\bot). \tag{3.22}$$

Therefore, since we assumed that $u(x_\bot) = 0$ and $u(x_\top) = 1$, then $u(z) = 2$.

Example: Suppose we have $x_\bot = 0$, $x_\top = 10$ and choose $r = 20$ and $R = 40$. We would ask the user to choose a probability p' such that they are indifferent between the gambles

$$[1 - p'; 0, p'40] \text{ and } [1 - p'; 10, p'20].$$

Suppose the user says $p' = 0.6$. We would then ask the user to choose an outcome z such that they are indifferent between the gambles

$$[0.4; 10, 0.6; 40] \text{ and } [0.4; z, 0.6; 20].$$

If $z > 20$ we would have to choose new values for r and R and repeat. □

Once we have found z, we can now solve Equation 3.16. While there is disagreement over the specific functional form of w, there is agreement that w is monotonically increasing [71]. That is, increasing the probability of an outcome will never cause a person to assign less weight to that outcome. As a result,

$$\frac{w(p)}{w(1-p)} \tag{3.23}$$

is also a monotonically increasing function with respect to p. This monotonicity means that we can do a binary search for p^*. Since for $p = 0$, Equation 3.23 equals 0 and as p approaches 1, Equation 3.23 approaches infinity, and Equation 3.23 is continuous, by the Intermediate Value Theorem, p^* will always exist [56].

Let $p^*_{\min}$ and $p^*_{\max}$ be the minimum and maximum values possible for p^*, respectively. We start with $p^*_{\min} = 0$ and $p^*_{\max} = 1$. Our estimate of p^* is p^*_{binary} which will always be equal to $(p^*_{\min} + p^*_{\max})/2$. The binary search will be based on asking the user if they prefer the decision

$$[1 - p^*_{binary}; x_\bot, p^*_{binary}; z] \tag{3.24}$$

or the decision

$$[p^*_{binary}; x_\bot, 1 - p^*_{binary}; x_\top]. \tag{3.25}$$

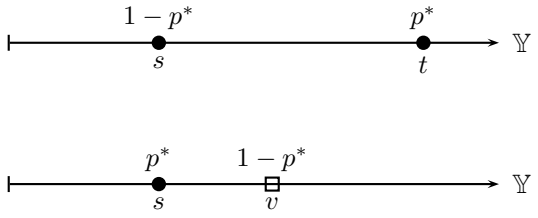


Figure 3.3: *A graphical representation of an outcome query. The user is asked to provide an outcome v such that they are indifferent between these two decisions. We can then infer that $u(v) = \frac{u(s)+u(t)}{2}$.*

If the user prefers the decision in Equation 3.24, then

$$\begin{aligned} w(p^*_{binary})u(z) + [1 - w(p^*_{binary})]u(x_\perp) &> w(1-p^*_{binary})u(x_\top) + [1 - w(1-p^*_{binary})]u(x_\perp), \\ w(p^*_{binary})u(z) &> w(1-p^*_{binary})u(x_\top), \\ \frac{w(p^*_{binary})}{w(1-p^*_{binary})} &> \frac{u(x_\top)}{u(z)}, \\ \Rightarrow \frac{w(p^*_{binary})}{w(1-p^*_{binary})} &> \frac{1}{2}. \end{aligned} \tag{3.26}$$

Therefore, our estimate of p^* is too high. In this case, we set $p^*_{\max} = p^*_{binary}$ and repeat this process. If the user prefers the decision in 3.25, by analogous reasoning, our estimate of p^* is too low. In this case, we set $p^*_{\min} = p^*_{binary}$ and repeat. In Section 3.5.1, we discuss how accurate the bounds on p^* need to be.

Example: Suppose our current bounds for p^* are $[p^*_{\min}, p^*_{\max}] = [0.25, 0.5]$. For our next iteration, we choose $p^*_{binary} = 0.375$. We ask the user if they prefer the decision $[0.625; x_\perp, 0.375; z]$ over the decision $[0.375; x_\perp, 0.675; x_\top]$. If the user prefers the first decision, we set $[p^*_{\min}, p^*_{\max}] = [0.25, 0.375]$ and otherwise we set $[p^*_{\min}, p^*_{\max}] = [0.375, 0.5]$. □

3.3.3 Outcome queries

Once we know p^*, we stop asking configuration queries and start asking outcome queries. An outcome query consists of asking the user to choose an outcome $v \in Y$ such that they are indifferent between the two decisions

$$[1-p^*; s, p^*; t] \text{ and } [p^*; s, 1-p^*; v], \tag{3.27}$$

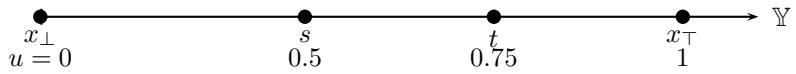


Figure 3.4: *A possible setting for an outcome query where we already know the utility values for the outcomes $x_\perp, s, t$, and $x_\top$. The outcomes s and t are not necessarily in $\mathbb{X}$.*

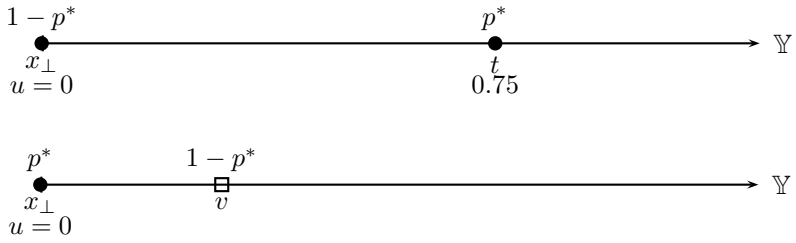


Figure 3.5: *A possible outcome query based on the known utility values from Figure 3.4.*

for outcomes $s, t \in \mathbb{Y}$. A graphical representation of this query is shown in Figure 3.3. From this indifference, we can infer that

$$\begin{aligned}
&w(1-p^*)u(v) + [w(1) - w(1-p^*)]u(s) \\
&\qquad = w(p^*)u(t) + [w(1) - w(p^*)]u(s) \\
&\Rightarrow w(1-p^*)u(v) - w(1-p^*)u(s) = w(p^*)u(t) - w(p^*)u(s) \\
&\Rightarrow u(v) = u(s) + \frac{w(p^*)}{w(1-p^*)}(u(t) - u(s)) \qquad (3.28) \\
&\qquad = u(s) + \frac{1}{2}(u(t) - u(s)) \qquad (3.29) \\
&\qquad = \frac{u(s) + u(t)}{2}.
\end{aligned}$$

Equations 3.28 and 3.29 show where we are able to use our knowledge about the user's probability weighting to remove its effect. Therefore, if we know $u(s)$ and $u(t)$, we now know $u(v)$. Initially, the only outcomes we know the utility of are $x_\perp$ and $x_\top$. Each time we query the user, we are able to use v in future querying. This means that, for the most part, s and t will not be members of $\mathbb{X}$.

Example: Suppose we know the 4 utility values $\{u(x_\perp) = 0, u(s) = 0.5, u(t) = 0.75, u(x_\top) = 1\}$. These values are illustrated in Figure 3.4. There are 6 different out-

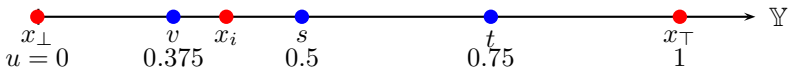


Figure 3.6: *An example of using our updated utility constraints, based on the results from the outcome query in Figure 3.5. Before the outcome query,* $C_{x_i} = [0, 0.5]$. *After the query, assuming monotonicity, we know that* $C_{x_i} = [0.375, 0.5]$.

come queries we could ask the user, one for each pair of outcomes whose utilities we know. For this example, we decide to query the user using the outcomes $x_\perp$ and t. Therefore, we ask the user to give an outcome $v \in Y$ such that they are indifferent between the two gambles

$$[1 - p^*; x_\perp, p^*; t] \text{ and } [p^*; x_\perp, 1 - p^*; v]. \tag{3.30}$$

A graphical representation of this query is shown in Figure 3.5. We now know that $u(v) = 0.375$. Suppose we have some outcome $x_i \in \mathbb{X}$ such that $x_\perp \prec x_i \prec s$, as shown in Figure 3.6. Before the outcome query, the utility constraints for $u(x_i)$ were $[0, 0.5]$. After the outcome query, the utility constraints can be updated to $[0.375, 0.5]$. Once we have updated all possible utility constraints for all outcomes in $\mathbb{X}$, we can recalculate the minimax regret and decide whether or not we need to query the user again. □

Example: To contrast outcome queries and SGQs, we revisit the used bookstore with SGQs example from Section 2.2. For simplicity, we assume that $p^* = 1/3$, *i.e.* EU holds. In the SGQ example, the student could sell the book for \$30 or consign it for \$80 with a 70% chance of the book being sold. The initial outcome query in this setting would ask the student to choose a price v such that they are indifferent between the gambles $[2/3; \$0, 1/3; \$80]$ and $[1/3; \$0, 2/3; v]$.

Assuming normalized utilities with $u(\$0) = 0$ and $u(\$80) = 1$, we know that $u(v) = 0.5$. Therefore, if $v < \$30$, we know that $u(\$30) > 0.5$ and otherwise, $u(\$30) \leq 0.5$. We will only know the exact value for $u(\$30)$ in in the exceptional case where $v = \$30$. Otherwise, we will have to use additional outcome queries to refine the utility constraints on v. □

The resulting preference elicitation algorithm using configuration queries and outcome queries is shown in Algorithm 2.

Algorithm 2 The preference elicitation algorithm for users with CPT-based preferences. This algorithm is based on the standard preference elicitation algorithm which is shown in Algorithm1.

while the range for p^* is too large **do**
  Query user using configuration query
  Use user's response to update range for possible values of p^*
end while
while regret given utility constraints C is greater than threshold **do**
  Query user using outcome query
  Use user's response to refine C
end while
Recommend optimal decision given C

3.3.4 The Gamble Equivalence Method and Expected Utility Theory

Probability weighting varies from person to person. Studies have shown that probability weighting is strongest in the people least familiar with a particular situation [17]. As people become experts in a field, they distort probabilities less. For these experts EUT may be a valid descriptive model. Since CPT is a generalization of EUT, GEM works regardless of whether or not people distort probabilities.

Although outcome queries can be used in many places where SGQs cannot, outcome queries do impose a higher cognitive burden. SGQs require only a yes/no response while outcome queries require a specific outcome as the response. One possible solution is to implement outcome queries as a series of simpler queries. For example, we could ask the user "is v less than or equal to y_1," "is v less than or equal to y_2," etc. Each of these simplier queries has the same cognitive burden as a SGQ. The tradeoff is that the user might have to answer a large number of these queries and some error is introduced into our utility bounds. A similar system of elicitation was used to elicit peoples' preferences for intertemporal scenarios, showing that this is a feasible approach in practice [23].

Since outcome queries are only necessary when people distort probabilities, we can rely on SGQs if can show that a user is not distorting probabilities. Under EU, we can solve

for p^* in Equation 3.16 as

$$\begin{aligned}\frac{w(p^*)}{w(1-p^*)} &= \frac{p^*}{1-p^*} \\ \Rightarrow p^* &= 1/3.\end{aligned}$$

It might be possible for $p^* = 1/3$ even with probability weighting. To see if $p^* = 1/3$ with probability weighting, we consider the case where the weighting is given by Kahneman and Tversky's weighting function in Equation 3.9. We want to see if there exists any γ which results in $p^* = 1/3$;:

$$\begin{aligned}\frac{w(1/3)}{w(2/3)} &= \frac{\frac{(1/3)^\gamma}{((1/3)^\gamma+(1-1/3)^\gamma)^{1/\gamma}}}{\frac{(2/3)^\gamma}{((2/3)^\gamma+(1-2/3)^\gamma)^{1/\gamma}}} \\ &= \frac{(1/3)^\gamma}{(2/3)^\gamma} \\ &= (1/2)^\gamma.\end{aligned}$$

In this case $p^* = 1/3$ if and only if $\gamma = 1$. If $\gamma = 1$ then

$$\begin{aligned}\frac{(p)^\gamma}{((p)^\gamma + (1-p)^\gamma)^{1/\gamma}} &= \frac{(p)}{(p+(1-p))}, \\ &= p.\end{aligned}$$

Therefore, if $\gamma = 1$, there is no probability weighting. If we use Prelec's probability weighting function,

$$w(p) = e^{-(-\ln p)^\gamma}, \tag{3.31}$$

the analysis is slightly different;

$$\begin{aligned}\frac{w(1/3)}{w(2/3)} &= 1/2 \\ \Rightarrow \frac{e^{-(-\ln 1/3)^\gamma}}{e^{-(-\ln 2/3)^\gamma}} &= 1/2 \\ e^{(-\ln 2/3)^\gamma-(-\ln 1/3)^\gamma} &= 1/2 \\ \ln\left[e^{(-\ln 2/3)^\gamma-(-\ln 1/3)^\gamma}\right] &= \ln 1/2 \\ (-\ln 2/3)^\gamma - (-\ln 1/3)^\gamma &= \ln 1/2 \\ (-\ln 2/3)^\gamma &= (-\ln 1/3)^\gamma + \ln 1/2. \end{aligned} \tag{3.32}$$

The left side of Equation 3.32 is monotonically decreasing while the right side is monotonically increasing, which means that there is at most one solution. Since $\gamma = 1$ is a solution, it is the only solution. As with the Kahneman-Tversky weighting function, with the Prelec weighting function, $\gamma = 1$ implies no probability distortion. This means that $p^* = 1/3$ means that the user is not distorting probabilities at all. Thus, with both the Kahenman-Tversky weighting function and the Prelec weighting function, we have the best of both worlds. If $p^* = 1/3$, we can rely on just SGQs. If $p^* \neq 1/3$, we know that we must fall back on outcome queries. We would like to strengthen these results by considering non-parametric cases.

3.4 Outcome Query Selection Heuristics

Choosing the best outcome query to ask the user can be difficult. We are often faced with many choices of possible queries to ask. The number of possible queries grows quadratically with respect to the number of outcomes we know the utility of. When eliciting preferences from real people, we must be able to determine the next query quickly; this means that we must be able to judge the value of each query quickly. The value of a query may depend on how the user responds, though it can often be that in the worst case, the user's response will not help us reduce the minimax regret. Since we do not know what the user's response to a query will be, we are unable to judge how likely the worst-case outcome is. Additionally, a series of queries may be more valuable than the sum of the value of each individual query. For example, a query may not reduce the minimax regret but instead allow for future queries which may be able to reduce the minimax regret considerably.

In this section we discuss four heuristics for helping us determine which queries to ask. The first two methods, *Halve Largest Gap* and *Current Solution*, were proposed by Boutilier *et. al.* for use with SGQs [10]. We discuss how to adapt these two methods for use with outcome queries. The next two heuristics, *Minimize Most-Likely Regret* (MMLR) and *Minimize Expected Minimax Regret* (MEMR) were introduced by Hines and Larson specifically for use with outcome queries [29].

The MMLR and MEMR heuristics both rely on assumptions about the distrubitions of the user's preferences. This is a shift from the previous work in this chapter where we have made minimizing the assumptions about users a priority. There are several reasons for this. First, these assumptions do not affect the correctness of any minimax regret calculations. Instead, these assumptions only affect the efficiency of the elicitation process. Second, there are already elicitation heuristics which do not make any assumptions about the user's preferences; specifically the Halve Largest Gap and Current Solution heuristics. In order

to examine alternatives to these existing heuristics, we need to make some assumptions. Finally, our goal is to design heuristics which still be useful when our assumptions are close to being correct. Thus, our heuristics should be flexible enough to handle cases that are not completely what we expected. We examine the issue of flexibility in Section 3.5.2.

It is possible for different outcome queries to elicit the same response. For example, suppose we have the outcomes y_1, y_2, y_3 and y_4 with the utility values 0.25,0.5,0.75 and 1, respectively. Then the response to the outcome query based on y_1 and y_4 will be the outcome y_5 such that $u(y_5) = 0.625$. Similarly, the response to the outcome query based on y_2 and y_3 will also be y_5. There is no difference between such outcome queries, *e.g.* we would be indifferent between using an outcome query based on y_1 and y_4 versus one based on y_2 and y_3. However, this does mean that it is possible to choose many different outcome queries which elicit the same response. This repetition is known as *stalling*. We can avoid stalling by always checking to see if we already know the response to an outcome query. For example, before querying using y_2 and y_3, we would check to see if we already know the outcome with utility 0.625.

3.4.1 Halve Largest Gap and Current Solution Heuristics With Outcome Queries

While the Halve Largest Gap (HLG) and Current Solution (CS) heuristics (Chapter 2.2.3) were designed to help choose which SGQs to use, they can both be easily adapted to work for outcome queries.

The HLG heuristic focuses on the outcome with the largest utility gap, *i.e.* the size of the corresponding utility constraint. While the HLG originally used a SGQ to halve this utility gap, we can just as easily use an outcome query to do the same. For any utility constraint $[u_{\min}, u_{\max}]$ there will always exist at least one outcome query which will result in halving that utility constraint: if $u_{\min}^{-1}(x)$ is the outcome whose utility is $u_{\min}$ and $u_{\max}^{-1}(x)$ is similarly defined, then by Equation 3.29, the outcome query based on $u_{\min}^{-1}(x)$ and $u_{\max}^{-1}(x)$ will always halve the utility constraint.

3.4.2 Minimize Most-Likely Regret

The first new heuristic, *Minimize Most-Likely Regret* (MMLR), uses a parametric approach where we choose some model utility function to approximate the user's. Although any

model can work, for our work we use the function

$$u(x) = x^{\hat{\beta}}, \tag{3.33}$$

partially because of its simplicity and partially because of its common use in experimental setting [71]. We use least squares fitting to match the parameters of the model function as closely as possible to the observed data. In the case of Equation 3.33, we would choose a value for $\hat{\beta}$ which minimizes the error. We then use the model to estimate the most likely response to an outcome query, which then allows us to estimate the most likely resulting minimax regret. The process is repeated for every possible outcome query. We then choose the query which we estimate will result in the lowest most-likely minimax regret.

3.4.3 Minimize Expected Minimax Regret

Our second new heuristic is *Minimize Expected Minimax Regret* (MEMR). The MEMR is inspired by the MEI heuristic, reviewed in Section 2.2.3 [68]. MEMR estimates the expected minimax regret resulting from every possible outcome query and chooses the query which minimizes the expected minimax regret. The calculation of the expected minimax regret is based on two key ideas.

The first idea is that we calculate the minimax regret ignoring the monotonicity constraint. Since the monotonicity constraint only matters when the utility constraints are overlapping, MEMR should be especially well suited for later in the elicitation process when the utility constraints are relatively small and not overlapping. Ignoring the monotonicity constraints makes calculating the pairwise maximum regret much easier. The PMR between decisions d_i and d_j is now calculated as

$$= \sum_{x \in X} (\Pr_{d_j}(x) - \Pr_{d_i}(x)) \cdot \begin{cases} u_{\max}(x) & \text{if } \Pr_{d_j}(x) \geq \Pr_{d_i}(x) \\ u_{\min}(x) & \text{otherwise.} \end{cases}$$

Therefore, to estimate the minimax regret after a query, we need to estimate the changes to $u_{\min}(x_i)$ and $u_{\max}(x_i)$ for all x_i. The outcome query (s, t), shown in Figure 3.7, will be able to update $u_{\min}(x_i)$ if and only if $u^{-1}_{\min}(x_i) \prec v \preceq x_i$, where v is the user's response. To estimate the probability of this occurring, we assume that the probability density of v is uniform between s and t. The MEI heuristic discussed in Section 2.2.3 also assumes a uniform distribution [68].

Figure 3.7: *A possible outcome query* (s, t) *that may be able to update either* $u_{\min}(x_i)$ *or* $u_{\max}(x_i)$.

With the assumption of uniform probability, the probability of $u_{\min}(x_i)$ being updated is

$$\frac{|x_i - u_{\min}^{-1}(x_i)|}{|t - s|}. \tag{3.34}$$

If $u_{\min}(x_i)$ is updated, we need to estimate by how much. We assume that the user's utility function is linear between $u_{\min}^{-1}(x_i)$ and $u_{\max}^{-1}(x_i)$. This becomes a reasonable assumption as the size of the utility constraints gets smaller since it has been observed that people's utility values tend to be relatively linear over small ranges [50]. Under this assumption, between $u_{\min}^{-1}(x_i)$ and $u_{\max}^{-1}(x_i)$, the slope of u is

$$\frac{u_{\max}(x_i) - u_{\min}(x_i)}{|u_{\max}^{-1}(x_i) - u_{\min}^{-1}(x_i)|}.$$

With the assumption of a uniform distribution for v, the expected value of v is

$$\left(\frac{|x_i - u_{\min}^{-1}(x_i)|}{2}\right).$$

Therefore, if $u_{\min}(x_i)$ is updated, the expected change in $u_{\min}(x_i)$ is

$$\left(\frac{|x_i - u_{\min}^{-1}(x_i)|}{2}\right)\left(\frac{u_{\max}(x_i) - u_{\min}(x_i)}{|u_{\max}^{-1}(x_i) - u_{\min}^{-1}(x_i)|}\right). \tag{3.35}$$

The overall expected change to $u_{\min}(x_i)$, $\Delta(u_{\min}(x_i))$, is given by Equation 3.34 multiplied by Equation 3.35. We can calculate $\Delta(u_{\max}(x_i))$, the expected change to $u_{\max}(x_i)$, in an analogous manner. For any two decisions d_i and d_j, the expected change to the PMR between those two decisions is

$$\Delta PMR(d_i, d_j) = \sum_{x \in X} (p_{d_j}(x) - p_{d_i}(x)) \cdot \begin{cases} \Delta(u_{\max}(x)) & \text{if } p_{d_j}(x) \geq p_{d_i}(x) \\ \Delta(u_{\min}(x)) & \text{otherwise.} \end{cases}$$

For each possible outcome query, we calculate the overall ΔPMR. This allows us to estimate the expected PMR resulting from any query. We then choose the outcome query which gives the lowest expected PMR.

3.5 Experimental Results

In this section, we present experimental results related to work done in this chapter. We examine the performance of both configuration queries and outcome queries.

3.5.1 Configuration Queries

The goal of configuration queries is to find the probability p^* which solves Equation 3.16. In Section 3.3.2 we presented a binary search method for finding lower and upper bounds, $p^*_{\min}$ and $p^*_{\max}$, on p^*. While we can use the binary search to find arbitrarily accurate bounds on p^*, since this is a search over the real numbers we will never know the exact value of p^*. However, at some point, our estimate of p^* will be accurate enough that we can behave as if we know the exact value. For example, in our experiments, we used the binary search for 10 iterations which meant that our estimate of p^* was within plus or minus 2^{-11} of the actual value. With this estimate, all of our utility constraint and regret calculations were correct. We confirmed this by always checking our utility constraints against the user's actual utility values.

Of course, in real life, we cannot check our constraints against the user's actual utility values. All we have to go on are our constraints on p^*. For example, if we could calculate

$$\frac{w(p^*_{\min})}{w(1-p^*_{\min})}, \tag{3.36}$$

and

$$\frac{w(p^*_{\max})}{w(1-p^*_{\max})}, \tag{3.37}$$

then we would have a lower and upper bound, respectively, on

$$\frac{w(p^*_{binary})}{w(1-p^*_{binary})}, \tag{3.38}$$

where p^*_{binary} is our estimate of p^*. If these bounds were too coarse, we could simply ask the user another configuration query. Since we have not made any assumptions about w, though, we cannot calculate Equations 3.36 and 3.37.

We decided to investigate the behaviour of Equation 3.38 as a function of the number of configuration queries. Since no weighting function has been shown to be overly accurate

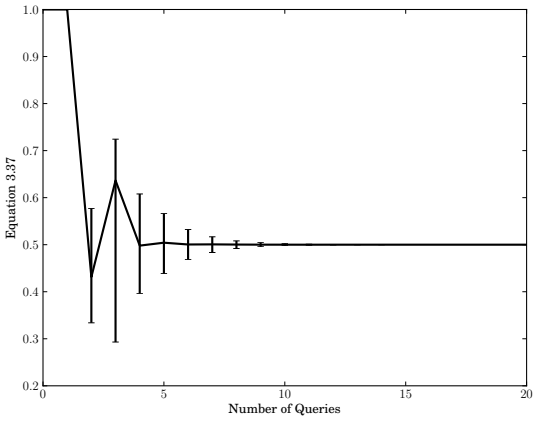


Figure 3.8: *The mean value for Equation 3.38 as a function of the number of configuration queries for simulated users using the Kahneman-Tversky weighting function, with γ chosen uniformly at random between 0 and 1. The range of minimum to maximum value is also shown.*

at modeling experimental results, we examined several different weighting functions and many different parameter values.

We first investigated the Kahneman-Tversky (KT) probability weighting function, shown in Equation 3.9. The KT weighting function has one parameter, γ, which has been shown to have a mean value of 0.71 and a standard deviation of 0.1. To provide the most robust results possible, we simulated 500 users with γ values chosen uniformly at random between 0 and 1. Figure 3.8 shows the mean value for Equation 3.38 after each configuration query as well as the minimum and maximum values. After 10 configuration queries, the mean value is equal to 0.50003 and the standard deviation is 0.001. For all 500 users, Equation 3.38 was within plus or minus 0.002 of 0.5. This means that after 10 queries, for all practical purposes, Equation 3.38 is equal to 0.5 which means we can assume that our estimate of p^* is correct.

We next investigated the Prelec probability weighting function, shown in Equation 3.10. The Prelec weighting function is also a function of γ, which has been experimentally shown to have a mean of 0.74 and a standard deviation of 0.14. Again, to provide the most robust results possible, we simulated 500 users with γ values chosen uniformly at random between 0.1 and 1. (Values of γ less than 0.1 resulted in overflow errors.) Figure 3.9 shows the mean value for Equation 3.38 after each configuration query as well as minimum and maximum values. After 10 configuration queries, the mean value is equal to 0.5000 and the standard

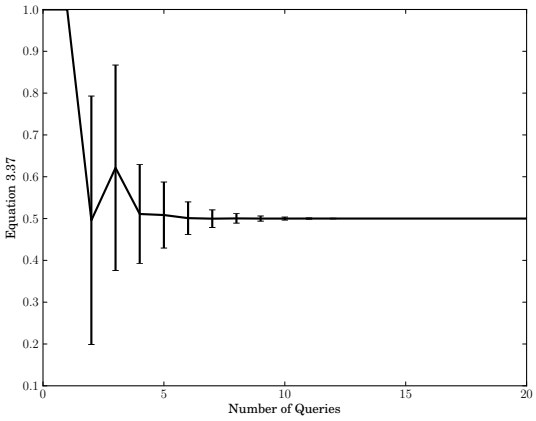


Figure 3.9: *The mean value for Equation 3.38 as a function of the number of configuration queries for simulated users using the Prelec weighting function, with γ chosen uniformly at random between 0.1 and 1. The range of minimum to maximum value is also shown.*

deviation is 0.002. The maximum and minimum values over all 500 users were within 0.02 of 0.5. Again this means that for most purposes, we can safely assume that Equation 3.38 is equal to 0.5 after 10 queries.

While the KT weighting function and the Prelec weighting function are the two most commonly-used weighting functions, neither have been very successful at modeling experimental data. To test the robustness of configuration queries, we need to investigate weighting functions which are substantially different. The only assumption configuration queries make about weighting functions is that they are monotonically increasing. The KT and Prelec weighting functions impose additional structure; specifically, the lower probabilities are over-weighted and higher probabilities are under-weighted. While all three of these characteristics are supported by experimental results, there is disagreement about the exact nature of the over-and under-weighting. Monotonicity appears to be the only characteristic of probability weighting that is generally agreed upon. Therefore, we next experimented with weighting functions that are monotonically increasing but do not necessarily have any of the over-or under-weighting characteristics.

The weighting function we chose to examine was a "quasi-uniformly" random weighting function. An example is shown in Figure 3.10. This weighting function chooses a finite set of probabilities and for each of those probabilities chooses the weighting uniformly at random (with the resulting weights sorted so monotonicity is maintained). Between any two of these points, the weight is given by a line connecting these points. With this

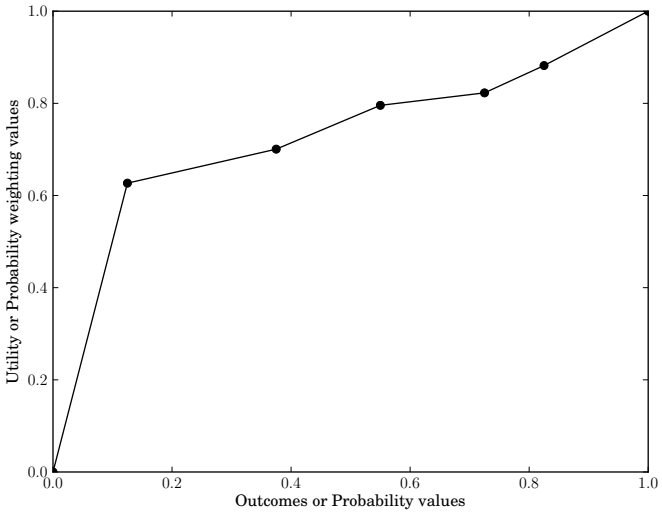


Figure 3.10: *An example of a quasi-uniformly random function. Quasi-uniformly random functions can be used as either probability weighting functions or utility functions. For a probability weighting function, we choose a fixed set of probability points and choose the corresponding weighting uniformly at random. These points are represented as dots in the above figure. The rest of the weighting values are created by connecting the dots with lines. For a quasi-uniformly random utility function, the dots represent the utility values for outcomes in* $\mathbb{X}$*, which are chosen uniformly at random. The utility values for the outcomes* $\mathbb{Y}$ *are then created by connecting the dots.*

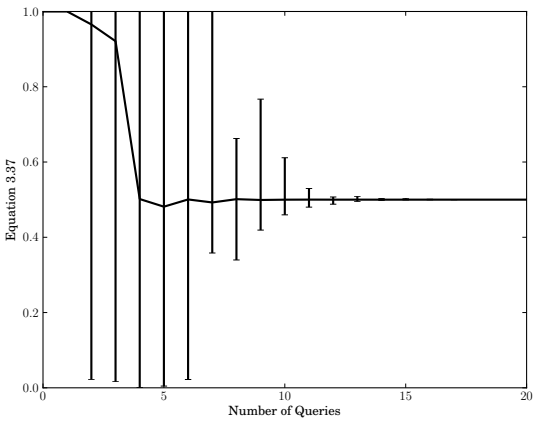


Figure 3.11: *The mean value for Equation 3.38 as a function of the number of configuration queries for simulated users using a quasi-uniformly random weighting function. The range of minimum to maximum value is also shown.*

type of function, it is possible for lower probability values to be underweighted and high probabilities to be overweighted. Hence, this function is fundamentally different than either the KT or Prelec weighting function. Figure 3.11 shows the mean values for Equation 3.38 as a function of the number of configuration queries, as well as the minimum and maximum values. After 10 configuration queries, the mean value is 0.5000 and the standard deviation is 0.01. This shows that, even with a fundamentally different type of probability weighting function, after 10 configuration queries, our estimate of p^* is, on average, close enough to being correct. However, the extreme minimum and maximum lie only within 0.11 of 0.5. This means that in the worst case, it may take 2 or 3 additional queries to find a sufficiently accurate estimate of p^*.

These results show that in reasonable cases, *i.e.* probability weightings with some resemblance to experimental results, configuration queries are able to find a sufficiently accurate estimate of p^* after 10 queries. Even with extreme probability weightings, we only need an additional two or three queries. Hence for all of the following experiments with outcome queries, we first used 10 configuration queries to estimate p^*.

3.5.2 Outcome Queries

Experimental Setup

For our experiments, we simulated processing 100 people. Each of these people were faced with 50 outcomes and 25 decisions. The outcomes and decisions were different for every person. We studied two different types of utility functions. The first utility function,

$$u(x) = x^{0.88}, \tag{3.39}$$

is a commonly-used function [41]. The second type of utility function we examined is a *quasi-uniformly random* utility function. For outcomes in $\mathbb{X}$ we chose the utility values uniformly at random between 0 and 1 and sorted the values to ensure monotonicity. Since outcome queries require users' preferences to be defined for any outcome in $\mathbb{Y}$, we defined the overall utility function to be a piece wise-linear function of the form

$$u(y) = my + b \tag{3.40}$$

where

$$m = \frac{u(x_{i+1}) - u(x_i)}{x_{i+1} - x_i}, \tag{3.41}$$

and

$$b = u(x_i) - mx_i, \tag{3.42}$$

for $x_i \prec y \prec x_{i+1}$. An example of this utility function is shown in Figure 3.10.

The experimental results in Section 3.5.1 show that configuration queries are robust enough to handle fundamentally different types of probability weighting functions. For simplicity in these experiments, we chose to use the KT probability weighting function with $\gamma = 0.71$.

We created two types of scenarios. In the first, both the outcomes and the decisions were picked uniformly at random. We then tried to create "harder" scenarios which maximized the initial minimax regret. For these scenarios, we first chose the outcomes uniformly at random, then decisions were created sequentially. For the first decision, d_1, we picked one outcome x at random and set $\Pr_{d_1}(x) = 1$ and all other values were set to zero.[3] For decision d_i, $i = 2, \ldots, 25$, we created ten decisions at random and picked the decision which maximized the minimax regret over the outcomes $d_1, \ldots, d_i$. Each user was queried until the minimax regret was at most 0.01. We tested four different elicitation heuristics. Our benchmark was the HLG heuristic. We also tested the CS, MMLR and MEMR heuristics.

[3]We found that choosing d_1 completely at random resulted in a lower overall minimax regret.

Description			Number of Queries	
Scenario type	Utility values	Elicitation heuristic	Mean	Median
Random	Power	HLG	18.23	15
Hard	Power	HLG	14.81	13
Random	Power	CS	13.46	12
Random	Power	MMLR	10.3	9.5
Random	Power + quasi-uniform	MMLR	11.74	9.5
Random	Power	MEMR	15.46	14
Random	Quasi-uniform	MEMR	16.75	16
Random	Power + quasi-uniform	MEMR	17.27	14

Table 3.3: *A summary of the experimental results presented in this section. Quasi-uniform utility values means utility values are chosen quasi-uniformly at random.*

Experimental Results

Our first simulation was created as a benchmark for all subsequent simulations. We created the outcomes and decisions at random. Our benchmark elicitation heuristic was HLG. The minimax and actual regret over 50 queries is shown in Figure 3.12 and summarized in Table 3.3. The mean and median number of queries are 18.23 and 15, respectively. This indicates that there were a few especially difficult scenarios; in fact, the mean and median number of the number of queries for the ten most difficult users using HLG were 40 and 40.5 respectively. The mean starting minimax regret is roughly 0.1. This is relatively low compared to the theoretical maximum minimax regret of 0.5 (assuming monotonicity). We also see that actual regret is considerably lower than the minimax regret. In many cases, the actual regret is 0 while the minimax regret is much higher. This means we can often pick the optimal decision without knowing it. These results also show that while the minimax regret can never increase, the actual regret can.

We repeated the benchmark simulation but this time chose the decisions to maximize the initial minimax regret. The results are shown in Figure 3.13 and summarized in Table 3.3. The average initial minimax regret is now over 0.25. While the actual regret remains low, we see that the highest average actual regret in Figure 3.13 is slightly higher than the highest average actual regret in Figure 3.12. This shows that we were successful at creating scenarios with a high initial minimax regret, and that, at least by one measure, these scenarios were more difficult than our benchmark scenarios. However, the mean and median number of of queries needed to process each user fell to 14.81 and 13, respectively. This indicates that a high initial minimax regret may actually be a poor indicator of a

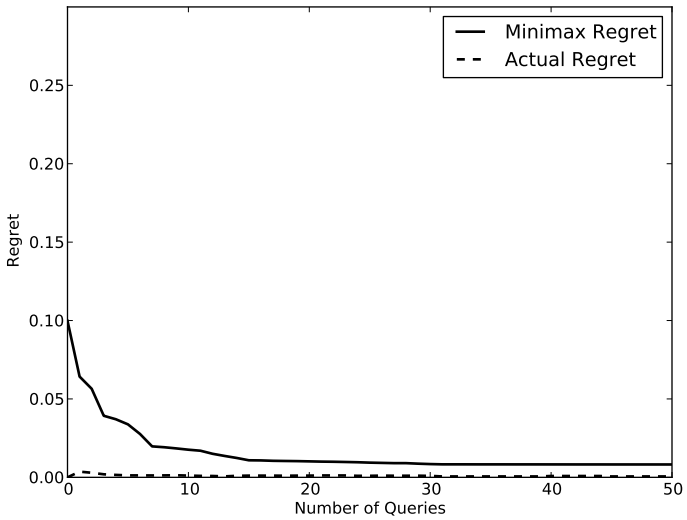


Figure 3.12: *The minimax and actual regret over 50 queries for users facing 50 outcomes and 25 decisions, chosen at random. The elicitation heuristic used was HLG.*

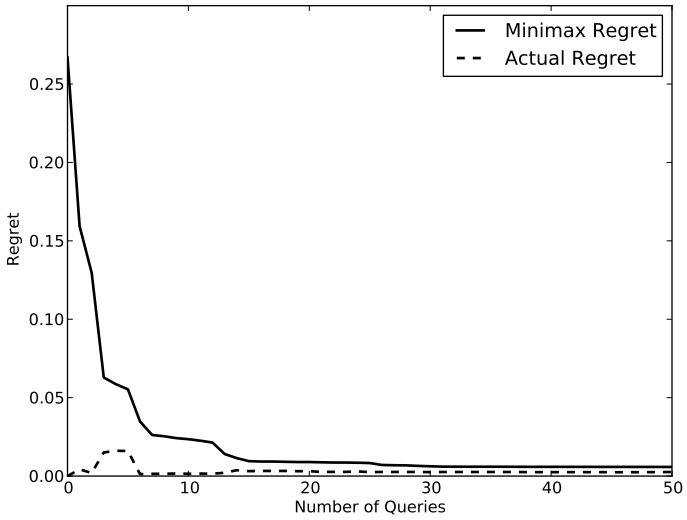


Figure 3.13: *The minimax and actual regret over 50 queries for users facing 50 outcomes and 25 decisions. The decisions were chosen to maximize the initial minimax regret. The elicitation heuristic used is HLG.*

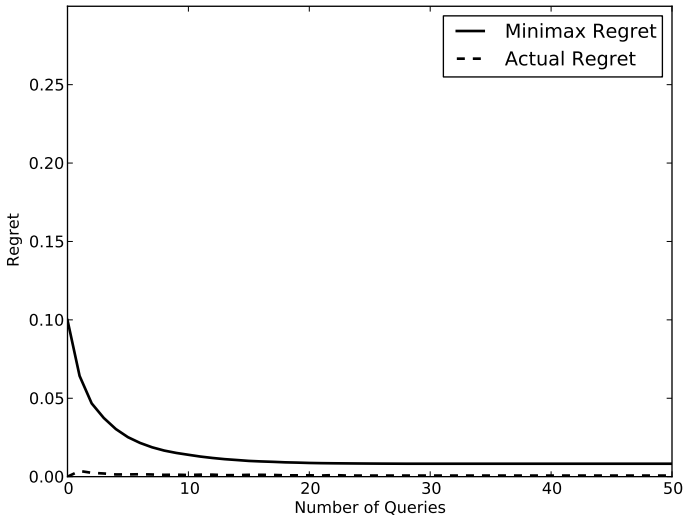


Figure 3.14: *The minimax and actual regret over 50 queries for users facing 50 outcomes and 25 decisions, chosen at random. The elicitation heuristic used is CS.*

scenario's overall difficulty. Therefore, for the rest of the experiments we created decisions at random.

In our next experiment, we studied the effectiveness of the CS elicitation heuristic. The results are shown in Figure 3.14 and summarized in Table 3.3. The mean and median number of queries needed to process each user, compared to our benchmark of HLG with randomly-generated decisions, decreased to 13.46 and 12, respectively. This was a decrease in the mean and median number of queries needed of 26% and 20%, respectively. As discussed in Section 2.2.3, CS does not offer the same theoretical gaurantees as HLG, in the worst case, using CS can result in an increase in the number of queries needed. We next checked to see how often this was the case for our results. Out of 100 users, there were only 5 users for whom using the CS heuristic resulted in an increase in the number of queries compared to using the HLG heuristic. For each of these users, the CS heuristic only resulted in one additional query. There were an additional 8 users that the HLG and CS heuristics processed in the same number of queries. Finally, we were interested in seeing how well the CS heuristic performed for the more difficult users. With CS, the mean and median number of queries for the ten most difficult users were 26 and 20, respectively. This was a decrease of 35% and 50%, respectively and shows that CS is able to handle the more difficult users efficiently.

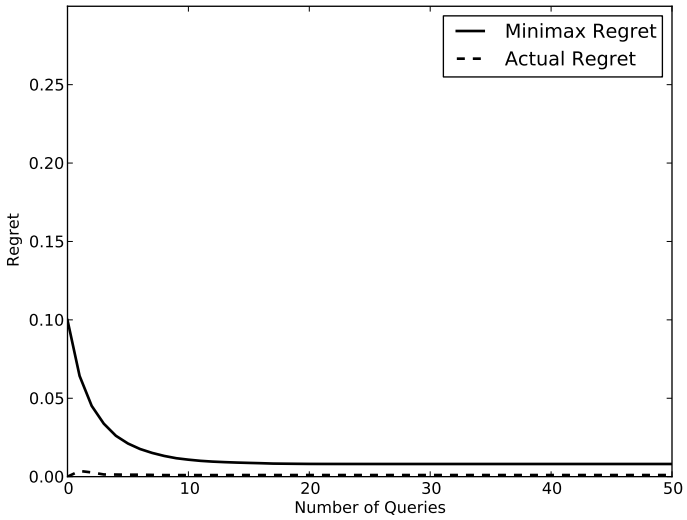


Figure 3.15: *The minimax and actual regret over 50 queries for users facing 50 outcomes and 25 decisions, chosen at random. The elicitation heuristic used is MMLR.*

In our next experiment, we studied the effectiveness of our MMLR elicitation heuristic. The results are shown in Figure 3.15 and summarized in Table 3.3. The mean and median number of queries decreases to 10.3 and 9.5, respectively. However, these results are highly dependent on the users' utility function having a similar shape to Equation 3.33.

To test the robustness of MMLR, we experimented with users with different types of utility values. If users have quasi-uniformly random utility values, MMLR was unable to process any users, even after 100 queries. We next experimented with a combination of the power utility function and the quasi-uniformly random utility function. For this combined utility function, we chose the utility values for outcomes in $\mathbb{X}$ using Equation 3.39 and for outcomes in $\mathbb{Y}$ using Equations 3.40 through 3.42. The results are shown in Figure 3.16 and summarized in Table 3.3. The mean and median number of queries are 11.74 and 9.5 queries, respectively. While these results are not as good as MMLR with a power utility function, they still outperform the CS heuristic.

This suggests that while in the worst case MMLR is not useful, in practice MMLR could be very useful. The law of diminishing returns, a common property of peoples' utility values, implies the concavity of utility functions. Thus, while peoples' utility functions may not perfectly match the the form of Equation 3.33, MMLR may be useful for many peoples'

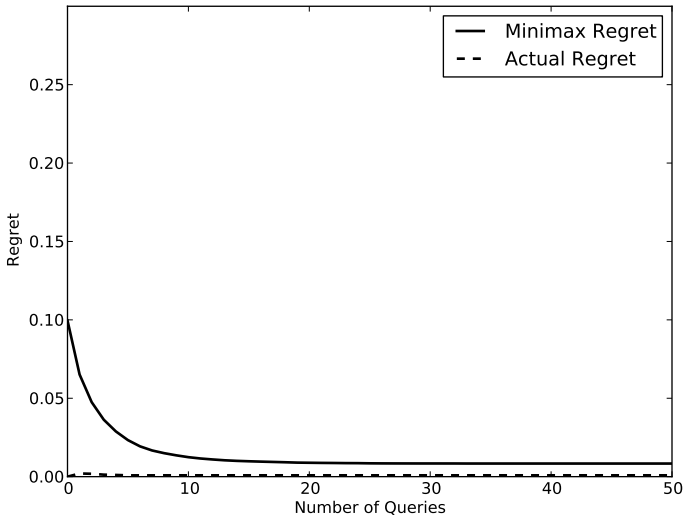


Figure 3.16: *The minimax and actual regret over 50 queries for users facing 50 outcomes and 25 decisions, chosen at random. The elicitation heuristic used is MMLR. Users' utility functions were based on a combination of a power utility function and a quasi-uniformly random utility function. The results are very similar to the results in Figure 3.15, where users' utility functions were purely a power utility function.*

actual utility values.

We next tested the MEMR heuristic. The results for a power utility function are shown in Figure 3.17 and summarized in Table 3.3. The mean and median number of queries were 15.46 and 14, respectively. While this is definitely an improvement over HLG, the MEMR heuristic was not as efficient as the CS or MMLI heuristics. We then used the MEMR heuristic to process users with quasi-uniformly random utility values. In this case, the mean and median number of queries increased to 16.75 and 16, respectively. We also used the MEMR heuristic to process users with the combined utility function previously mentioned. In this case, the mean and median number of queries were 17.27 and 14 respectively. This suggests that, while MEMR may not be as efficient as MMLI, MEMR is a flexible heuristic and is able to deal with many different types of utility functions.

In theory, the MEMR and MMLR heuristics can choose an outcome query based on any two outcomes which we already know the utility of. In practice, however, they always recommend two adjacent outcomes, *i.e.* we do not know the utility values for any outcomes

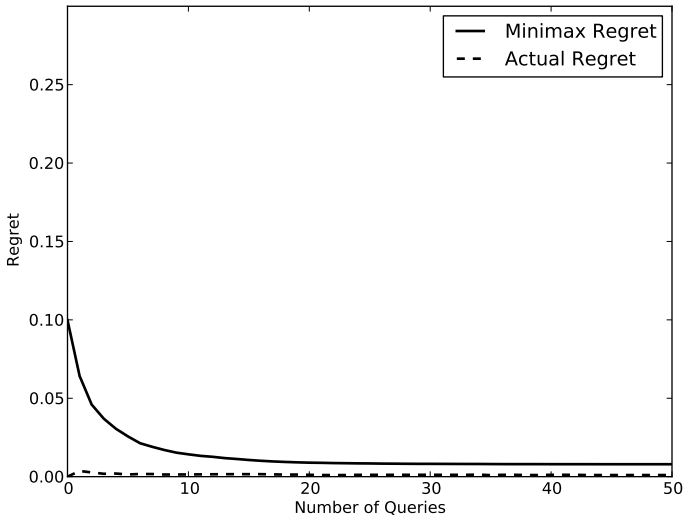


Figure 3.17: *The minimax and actual regret over 50 queries for users facing 50 outcomes and 25 decisions, chosen at random. The elicitation heuristic used is MEMR.*

between the two that were recommended. As a result, we can save considerable time by having MEMR and MMLR only search through adjacent outcomes.

3.6 Conclusion

In this chapter we examined whether standard gamble queries (SGQs) are compatible with cumulative prospect theory (CPT). We showed that, in the worst case, using SGQs when a user's preferences follow CPT can result in arbitrarily bad decisions. We also showed experimentally that in the average case, the error can be a concern. As a result, we proposed the Gamble Equivalence Method for preference elicitation which is compatible with CPT and allows us to efficiently reduce the minimax regret. We also developed two heuristics to help us choose the best query and discussed how to adapt two preexisting heuristics, originally developed for SGQs, to use with our method. Finally, we presented experimental results showing how these heuristics performed in simulated settings. When users' preferences were of the form $u(x) = x^{0.88}$, the MMLR heuristic easily outperformed both the HLG and CS heuristics. The MMLR heuristic also performed well when users had utility values close to $u(x) = x^{0.88}$. In the worst case, however, when users had utility

values chosen uniformly at random, MMLR was not able to process users. The MEMR heuristic was not as efficient as MMLR when when $u(x) = x^{0.88}$ but MEMR was still able to outperform HLG. The MEMR was also more flexible and was always able to process users. This suggests that in the best case MMLR is the best heuristic, but that MEMR has more flexibility.

Chapter 4

Probabilistic Models of Regret

Regret is the measure of error most commonly used in preference elicitation. In a sense, we may think of the goal of preference elicitation to reduce the regret as quickly as possible. The faster we reduce the regret, the less we have to bother the user with queries.

Since regret is a function of the user's utility values, which are private, the true regret is often not known. Instead, measures of regret used in preference elicitation must be approximations or bounds of the actual regret. Improving these approximations or bounds will allow for a faster and more efficient elicitation process.

In this chapter, we explore new ways of approximating and bounding the actual regret. Our focus is on situations where we are eliciting preferences from a group of users. In Section 4.1, we review the current methods for measuring regret. We introduce our new method in Section 4.2. In Section 4.3, we discuss different methods of optimization. Finally, in Section 4.4 we provide experimental results.

4.1 Current Measurements of Regret

Two measurements of regret currently in the literature used are expected regret and minimax regret. Both measurements have advantages and disadvantages.

Expected regret (ER) (Section 2.2.2) assumes that the current user's preferences are drawn from a known distribution over possible preferences. If we knew the user's exact utility values, then the expected regret from choosing the decision d instead of the decision d^*_{EU} which maximizes expected utility would be

$$r(d, u) = EU(d^*_{EU}, u) - EU(d, u).$$

If we only have $\Pr(u)$, the probability distribution from which the user's preferences were drawn, then we can calculate the ER as

$$ER(d, \Pr(u|C)) = \int [EU(d^*_{EU}(u), u) - EU(d, u)] \Pr(u|C) du,$$

where $d^*_{EU}(u)$ is the decision which maximizes expected utility given the utility values u. The advantage of expected regret is that we are dealing with expected cases instead of worst cases (as with minimax regret). Expected cases typically give lower regret estimations and allow us to process users with fewer queries. The disadvantage of expected regret is the strong prior knowledge requirement. The distribution is based on preferences elicited from previous users [19, 20]. This means that we cannot assume $\Pr(u)$ is known perfectly; any sample of users will be of finite size and any finite sample introduces the possibility of error [69]. We are unaware of any work dealing with the issue of error in ER calculations; previous work has assumed that $\Pr(u)$ is known perfectly [19, 20]. It is possible to have a sample size large enough that it might be reasonable to assume that any error is "more or less" non-existent. There are two factors which may limit the applicability of this assumption. First, we must have already processed a large number of users whom we know are drawn from the same distribution of preferences. This is not a reasonable assumption for a new "setting," *e.g.* if we are starting to help people create energy policies for the Smart Grid. Even a database of previously-processed users may not help since the information in it may be outdated or not fully relevant to the decision at hand. The second factor is the quality of the utility constraints we have for each processed user. Anything less than knowing the exact utility values may introduce error into our calculations. If the utility constraints prove to be inexact, we cannot go back and "reprocess" users; they may have moved or they may be unwilling to give more of their time.

Minimax regret avoids the prior knowledge assumption used in expected regret calculations by not making any assumption about the user's preferences. The tradeoff is that minimax regret focuses on the worst-case outcomes which can result in an overly cautious upper bound on the actual regret. This caution can result in a need for additional queries to process each user. We could decrease the number of queries needed if we knew by how much the minimax regret overestimated the actual regret. The difficulty is that the actual regret may be equal to the minimax regret or it could be significantly less.

We first show that it is always possible for the minimax regret and the actual regret to be equal.

Lemma 6. *For any set of utility constraints C, it is always possible for the actual regret to equal the minimax regret, i.e.*

$$R(d^*_{MMR}, u) = MMR(C),$$

where d^*_{MMR} is the minimax-optimal decision, i.e. the decision which minimizes the maximum possible regret.

Proof. The *adversarial decision* d^a is the decision which maximizes pairwise maximum regret with respect to d^*, *i.e.*

$$d^a = \arg\max_{d \in D} PMR(d^*_{MMR}, d, C). \tag{4.1}$$

By the definition of maximum regret in Equation 2.14 and Equation 4.1,

$$R(d^*_{MMR}, u) = PR(d^*_{MMR}, d^a, u),$$

where $PR(d, d', u)$ is the pairwise regret from choosing decision d instead of d'. The *adversarial utility* u^a is the set of utility values which maximizes the pairwise regret between d^* and d^a, *i.e*

$$u^a = \arg\max_{u \in C} \left\{ EU(d^a, u) - EU(d^*, u) \right\}. \tag{4.2}$$

Therefore, if the user's actual utility values are u^a,

$$PMR(d^*_{MMR}, d^a, C) = PR(d^*_{MMR}, d^a, u^a). \tag{4.3}$$

If Equation 4.3 holds,

$$\begin{aligned} MMR(C) &= MR(d^*_{MMR}, C) \\ &= \max_d PMR(d^*_{MMR}, d, C) \\ &= PR(d^*_{MMR}, d^a, u) \\ &= R(d^*_{MMR}, u). \end{aligned}$$

□

We next investigated cases where there is a considerable difference between the minimax and actual regret. To do so, we created 500 users modelled with the utility function

$$u(x) = x^\beta, \tag{4.4}$$

with β being chosen uniformly at random between 0.5 and 1. Each user faced the same 20 outcomes and we created 10 decisions, uniformly at random, for each user. We then compared the initial average actual and minimax regret (*i.e.*, the actual and minimax regret

Regret	Nonmonotonic	Monotonic
Minimax	0.451	0.123
Actual	0.052	0.008

Table 4.1: A comparison of the initial minimax and actual regret for users with and without the monotonicity constraint.

before any queries). The results are shown in the left column of Table 4.1. The average minimax regret was 0.451 while the average actual regret was only 0.052, a difference of almost an order of magnitude.

In order to improve the minimax regret, we need to make additional assumptions about the users' preferences. Since Equation 4.4 guarantees that the users' preferences will be monotonically increasing, we could try to decrease the minimax regret by imposing a monotonicity constraint on the users' utility values. The results from imposing this constraint are shown in the right-hand column of Table 4.1. With the monotonicity constraint, the minimax regret decreases to 0.123. Since our regret calculations change, the decisions we choose on behalf of the user may also change. In fact, Table 4.1 shows that by imposing a monotonicity constraint, we are also choosing better decisions on behalf of the user. In this case, the average actual regret decreases to 0.008. Thus, even with the monotonicity constraint, the minimax regret remains roughly an order of magnitude higher than the actual regret.

Therefore, the minimax regret may be equal to the actual regret, or there may be a magnitude of difference between them. Even if we know that the actual regret is less than the minimax regret, we need a quantitative measurement of the difference. For example, suppose we are in a situation where the minimax regret is 0.1. If the maximum actual regret we can tolerate is 0.01, can we stop querying the user? Extrapolating form the results in Table 4.1, the actual regret might be 0.0115 or it might be 0.006. In one case, we can stop querying the user and in the other case we cannot. Therefore, a more principled approach is needed.

4.2 Probabilistic Regret

In this section, we present *probabilistic regret*: a new way of approximating and bounding the actual regret. We have three goals with probabilistic regret. First, we want to provide a more accurate upper bound on the actual regret compared to minimax regret; in this sense, probabilistic regret is closer to expected regret. Second, we want to relax the prior

knowledge assumption needed for expected regret; in this sense, probabilistic regret is closer to minimax regret. Hence, probabilistic regret allows us to view regret as a spectrum between expected and minimax regret. The third goal is to provide flexibility. There will always be a balance between accuracy and prior knowledge. Our aim is to allow the controller (*i.e.* whoever is running the elicitation process) to be able to decide on that balance.

Probabilistic regret gives a probabilistic upper bound on the actual regret from choosing a decision. For example, we would like to be able to say that there is an 80% probability that choosing the decision d will give us an expected regret of at most 0.2. The advantage of probabilistic regret is that we can ignore the rare cases where the regret is especially high. For example, suppose the maximum regret from choosing d might be 0.3 but there is an 95% probability that the actual regret from choosing d is at most 0.1. By ignoring the other 5%, we can give a lower estimate of the regret.

We relax the prior knowledge assumption by assuming that $\Pr(u)$ is unknown. Instead, we assume that we are eliciting individual preferences sequentially from a group of users.[1] For example, we could be helping households in a specific city to each choose an optimal energy use policy. We assume that the preferences from each of these users are drawn i.i.d. from a single fixed but unknown distribution. This is the same setting as Bayesian learning but without the need of any prior knowledge or training data [44]. Probabilistic regret may benefit from prior knowledge, but it is never required. We assume that the possible outcomes are the same for all users. For technical simplicity, we assume that all users face the same set of possible decisions. These are both reasonable assumptions since we are helping a group of users with the same problem. Generalizing our approach so that users can have a different set of possible decisions requires only generalizing the notation.

For technical simplicity, the work in this chapter assumes that people follow expected utility theory. All of the results in this chapter are completely compatible with configuration and outcome queries. The methods only require a set of utility constraints from each other. We are indifferent as to whether these constraints were obtained by standard gamble queries or configuration and outcome queries. Hence, the results in this chapter are compatible with cumulative prospect theory.

Probabilistic regret is based on standard methods from non-parametric statistics. While there are many similarities between our approach and Bayesian reasoning, probabilistic regret does not use Bayesian methods. It is theoretically possible that Bayesian methods

[1] We do not require users to be processed completely in sequence: we do allow some users to be processed in parallel. What is important is that, for each user, we have a set of preferences from previous users to help us.

could provide an alternative to probabilistic regret. Therefore, we also discuss some possible complications a Bayesian approach might encounter and the advantages of our approach.

4.2.1 Probabilistic Regret

In this section, we introduce our initial definition of probabilistic regret. The initial definition assumes that $\Pr(u)$ is known. In subsequent sections, we discuss how to relax this assumption.

Given $\Pr(u)$, we first define the probability distribution function $r(d, d', \Pr(u))$ to be the distribution of pairwise regret values given $\Pr(u)$. We next define the *cumulative density function* (cdf), $F_{d,d'}(r)$, as

$$F_{d,d'}(r) := \Pr(r(d, d', \Pr(u)) \leq r),$$

i.e. $F_{d,d'}(r)$ is the probability that the pairwise regret is at most r. Throughout the remainder of this chapter, we will be using numerous variations on $F_{d,d'}(r)$ as well as several lower and upper bounds on $F_{d,d'}(r)$. While all of these functions are dependent on both d and d', to keep notation simple, we will avoid listing d and d', instead only writing $F(r)$, unless d and d' are especially relevant at that point.

Given a set of utility constraints C, we can make $r(d, d', \Pr(u))$ dependent on C, *i.e.* $r(d, d', \Pr(u|C))$ is the probability distribution function of the pairwise regret given C. Similarly, we can make $F(r)$ dependent on C, *i.e.*

$$F_C(r) := \Pr(r(d, d', \Pr(u|C)) \leq r),$$

that is, what do we expect the pairwise regret between decisions d and d' to be given the conditional probability distribution $\Pr(u|c)$. For a given probability p, we define the *probabilistic pairwise maximum regret* (PrPMR) as

$$PrPMR(d, d', C, p) := F_C^{-1}(p). \tag{4.5}$$

For example, if $PrPMR(d, d', C, 0.9) = 0.2$, then given the utility constraints C, with probability of error 0.1, the pairwise regret from choosing d instead of d' is at most 0.2. There is a strong relationship between $PrPMR$ and PMR. Specifically,

$$PrPMR(d, d', C, 1) \leq PMR(d, d', C).$$

That is, the pairwise maximum regret is an upper bound on the probabilistic pairwise maximum regret.

Using this definition, we define the *probabilistic maximum regret* (PrMR) as

$$PrMR(d, C, p) := \max_{d' \in \mathbb{D}} PrPMR(d, d', C, p). \tag{4.6}$$

We next define the *probabilistic minimax regret* (PrMMR) as

$$PrMMR(C, p) := \min_{d \in \mathbb{D}} PrMR(d, C, p). \tag{4.7}$$

Finally, we define the *probabilistic minimax optimal decision* d^* as

$$d^* := \arg\min_{d} PrMR(d, C, p). \tag{4.8}$$

We illustrate the meaning of the above terms through the use of the following examples. These examples also allow us to compare and contrast probabilistic regret with expected regret and minimax regret. All of the following examples assume independence between pairwise regret $F_{d,d',C}(r)$ for different tuples of d and d'.

Example: If we set $p = 1$, $PrMR(d, C, 1)$ is the guaranteed maximum regret for choosing decision d given utility constraints C. In this case, $PrMMR$ gives the best worst-case outcome. Note that, $PrMR(d, C, 1) = MR(d, C)$ may or may not hold. This is because F_C may be able to rule out certain preferences in C. □

Example: If $p = 0.5$, $PrMR(d, C, 0.5)$ is the median regret for choosing decision d given utility constraints C. Unless the median and mean are the same, median regret will be different than expected regret. If we want the mean regret, we must sample user preferences according to $F_C(r)$, which is the process used to calculate expected regret. □

Example: We can also choose values of p between 0.5 and 1. For example, $PrMR(d, C, 0.9)$, gives us the 90th percentile regret from choosing decision d given constraints C. Thus we can use probabilistic regret to achieve a balance between maximum regret and expected regret. □

Example We next consider PrMMR. If we set $p = 1$, $PrMMR(C, 1)$ is the best worst-case scenario given $\mathbb{D}$, utility constraints C and probability distribution $\Pr(u)$. This is different than minimax regret, since minimax regret does not depend on any probability distribution. This distinction is explored further in the experimental results section. If we now set $p = 0.5$, $PrMMR(C, 0.5)$ gives us the best median scenario. Again, this is similar but not identical to minimizing expected (mean) regret. □

If the pairwise regret from choosing decision d instead of d' is greater than $PrMR(d, d', C, p)$, we cannot offer any bound on what the regret might be (other than the

obvious trivial bound of 1.) This means that with probability $1 - p$, we have no idea how bad choosing decision d might be. For example, if we set $p = 0.99$, then with probability 0.01, we may experience a *black swan*, a rare but profound event [57]. Thus, we may have two different goals. The first goal is to minimize the regret for 99% of the users. The second goal is to minimize the probability of a black swan occurring. In this case, both our measure of optimality for choosing a decision and our measure of regret are based on aggregations of these two, possibly competing, goals. For example, the aggregate utility function could be based on lexicographical preferences. Thus, for technical simplicity, we will avoid such issues for now while still recognizing their importance.

4.2.2 Relaxing the Prior Knowledge Assumption

Our current definition of probabilistic regret assumes that $\Pr(u)$ is known, which is the same assumption made by expected regret. (Minimax regret assumes no prior knowledge.) In this section, we investigate how to relax this prior knowledge assumption. We examine two different approaches. The first method is to learn about $F(r)$ as we process each user. With the second method, we start with a set of hypotheses about what $F(r)$ could be. As we process each user, we are eventually able to reject the incorrect hypotheses. These methods may be thought of as complementary. With the first method, we assume nothing and eventually find the right answer. With the second method, we start by assuming anything is possible and eventually reject all of the incorrect assumptions.

Learning the Probability Distribution

Our first approach to relaxing the prior knowledge assumption is called *learnt PrMMR* (lPrMMR). The basic idea is to learn $F(r)$ as we process additional users. We develop an approximation of $F(r)$ that is initially crude, and with each additional user we process, the approximation gets closer to $F(r)$. The only assumption we make about $F(r)$ is that users' preferences are drawn i.i.d..

Suppose we have processed users up to, but not including, user i, and we have found the exact utility values for each of those users. We could then approximate $F(r)$ with an *empirical distribution function* (edf) $\hat{F}_i(r)$, defined as

$$\hat{F}_i(r) := \frac{1}{i-1} \sum_{j<i} I(r(d, d', u_j) \leq r), \tag{4.9}$$

where

$$I(A \leq B) := \begin{cases} 1 & \text{if } A \leq B \\ 0 & \text{otherwise,} \end{cases}$$

and u_j is the set of utility values for user j. For example, if $\hat{F}_i(0.2) = 0.9$, then for 90% of all $i-1$ previous users, the regret from choosing decision d instead of d' was at most 0.2. To help keep notation simple, we will not list i when discussing $\hat{F}_i(r)$ or any lower or upper bounds based on $\hat{F}_i(r)$, unless necessary.

After we have processed user i enough to obtain the utility constraints $C(x)$, we can improve the accuracy of Equation 4.9 by considering only those previously-processed users whose utility values are *compatible* with $C(x)$. A user j with utility values $u_j(x)$ is compatible with the utility constraints $C(x)$ if and only if

$$C_{\min}(x) \leq u_j(x) \leq C_{\max}(x) \text{ for all } x \in \mathbb{X}. \tag{4.10}$$

We define the set of users $S_i(C)$ to be all users processed before user i whose utility values satisfy Equation 4.10 given the utility constraints C. This allows us to refine Equation 4.9 as

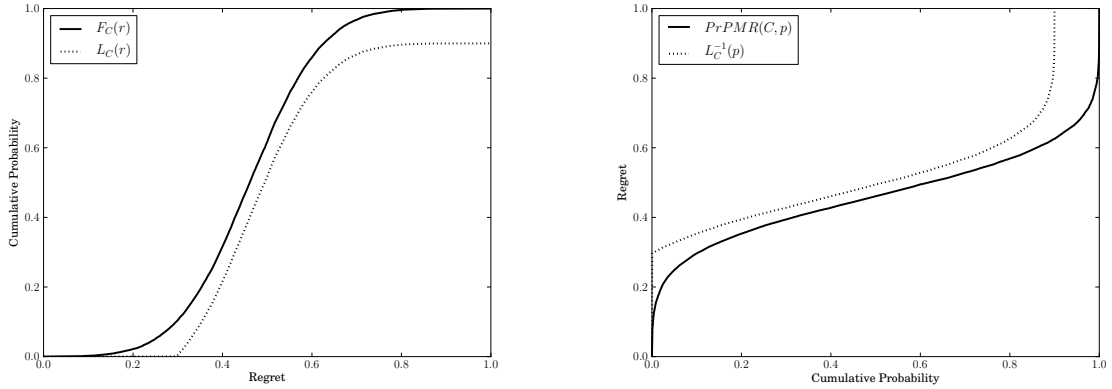
$$\hat{F}_C(r) := \frac{1}{|S_i(C)|} \sum_{j \in S_i(C)} I(r(d, d', u_j) \leq r). \tag{4.11}$$

There are two problems with Equation 4.11. First, since we do not know the exact utility values for any of the previous users, we cannot calculate $r(d, d', u_j)$ and therefore we cannot calculate Equation 4.11. The second problem is that $\hat{F}_C(r)$ is only an approximation of $F_C(r)$. We need to know how accurate this approximation is. For example, after processing 10 users, we would like to know what the probability is that the cdf and edf differ by at most 0.1. We solve both of these problems by finding three successive lower bounds to $F_C(r)$. As illustrated in Figure 4.1, finding a lower bound on $F_{d,d',C}(r)$ allows us to find an upper bound on $PrPMR(d, d', C, p)$ (denoted as $PrPMR(C, p)$ for brevity).

The first lower bound, $L_C^1(r)$, is due to the fact that since we do not know the exact utility values of any of the previous users, we cannot calculate $r(d, d', u_j)$. Instead, we rely on Equation 2.14 to provide an upper bound on $r(d, d', u_j)$. Thus, we relax Equation 4.9 to

$$\hat{F}_C(r) \geq L_C^1(r) := \frac{1}{|S_i(C)|} \sum_{j \in S_i(C)} I(PMR(d, d', C_j) \leq r). \tag{4.12}$$

For example, $L_C^1(0.3) = 0.2$ means that the probability of the pairwise regret from choosing decision d instead of d' being at most 0.3 is at least 20%. To account for the



(a) Cumulative Probability

(b) Inverse Cumulative Probability

Figure 4.1: *Example of how finding a lower bound can lead to finding an upper bound. In Figure 4.1(a) ,the solid line is an example cdf, $F_C(r)$, of the pairwise regret between decisions d and d'. The dashed line, $L(r)$, is a lower bound of $F_C(r)$. Since $PrPMR(C,p) = F_C^{-1}(p)$, Figure 4.1(b) is the inverse of Figure 4.1(a). We note that $L^{-1}(p)$ is an upper bound on $PrPMR(C,p)$.*

worst possible case, we take equality in Equation 4.12. An example of the relationship between $F_C(r)$, $\hat{F}_C(r)$ and $L_C^1(r)$ is shown in Figure 4.2.

Since we do not know u_j, we must also redefine compatibility in terms of C_j. Given the utility constraint C_j, a user is compatible with the utility constraints C_i if and only if

$$C^i_{\min}(x) \leq C^j_{\min}(x) \text{ and } C^j_{\max}(x) \leq C^i_{\max}(x) \text{ for all } x \in \mathbb{X}. \tag{4.13}$$

Figure 4.3 shows some examples of both compatible and incompatible constraints. As Figure 4.3 illustrates, there can be cases where a user would be compatible according to Equation 4.10 but not Equation 4.13.

Although $L_C^1(r)$ is a lower bound of $\hat{F}_C$, $L_C^1(r)$ is not necessarily a lower bound of F_C. As a result, we need to create two additional lower bounds, $L_C^2(r)$ and $L_C^3(r)$. We can measure our confidence that $L_C^1(r)$ is a lower bound of F_C by giving a *confidence band* around $L_C^1(r)$. A confidence band is a region around $L_C^1(r)$ which we are confident includes F_C. According to the Dvoretzky-Kiefer-Wolfowitz inequality [70], given a desired

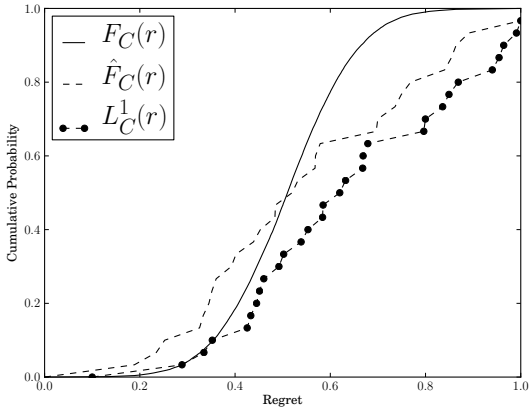


Figure 4.2: *An example of estimating the cdf for the regret of choosing decision d over d′. The actual cdf is $F_C(r)$. Our edf is $\hat{F}_C(r)$. Since we cannot calculate the actual regret values, we use Equation 4.12 to create $L^1_C(r)$ which is a lower bound of $\hat{F}_C(r)$.*

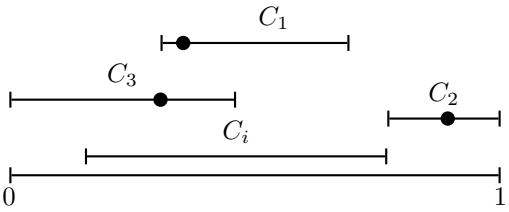


Figure 4.3: *A possible set of constraints for $u(x_1)$ where C_i is the constraint set for the current user i. The dots represent the actual utility values for users (unknown to the elicitation process). The constraint C_1 is compatible with C_i while C_2 is not. We consider C_3 incompatible with C_i, even though there is a chance that the two users have the same utility value.*

error value of

$$\epsilon := \sqrt{\frac{1}{2|S_i|}\log\left(\frac{2}{\alpha}\right)}, \tag{4.14}$$

for some chosen $0 < \alpha \leq 1$, we can construct the functions

$$L_C^2(r) := \max\{\hat{F}_C(r) - \epsilon, 0\}$$

$$U_C(r) := \min\{\hat{F}_C(r) + \epsilon, 1\}$$

such that

$$\Pr\left[L_C^2(r) \leq F_C(r) \leq U_C(r) \text{ for all } r\right] \geq 1 - \alpha. \tag{4.15}$$

We are interested in the implication that

$$\Pr\left[L_C^2(r) \leq F_C(r) \text{ for all } r\right] \geq 1 - \alpha.$$

However, since we cannot calculate $\hat{F}_C(r)$, we also cannot calculate $L_C^2(r)$. Instead, we consider another lower bound,

$$L_C^3(r) := \max\{L_C^1(r) - \epsilon, 0\}. \tag{4.16}$$

Since $L_C^1(r) \leq \hat{F}_C(r)$, it follows that

$$\Pr\left[L_C^3(r) \leq F_C(r) \text{ for all } r\right] \geq 1 - \alpha.$$

Therefore, the probability that $PMR(d, d', C) \leq r$ is at least $(1 - \alpha)L_C^3(r)$ (where L_C^3 is also a function of ϵ). An example showing $L_C^3(r)$ with respect to $F_C(r)$ and $\hat{F}_C(r)$ is shown in Figure 4.4.

With slight abuse of notation, we define $L_C^{-3}(p_{cd})$ to be the inverse of $L_C^3(r)$. This allows us to bound the PrPMR by

$$PrPMR(d, d', C, p_{cd}) \leq L_C^{-3}(p_{cd}) \tag{4.17}$$

for some given probability p_{cd}. That is, with probability $(1 - \alpha)p_{cd}$, the maximum regret from choosing decision d over d' is at most $L_C^{-3}(p_{cd})$. To account for the worst case, we assume equality in Equation 4.17. Therefore, we define the *learnt PrPMR* as

$$lPrPMR := L_C^{-3}(p_{cd}),$$

where p_{cd} is the cumulative probability chosen by the controller (*i.e.* who ever is runnig the elicitation process.) We let

$$p_{err} := 1 - (1 - \alpha)p_{cd} \tag{4.18}$$

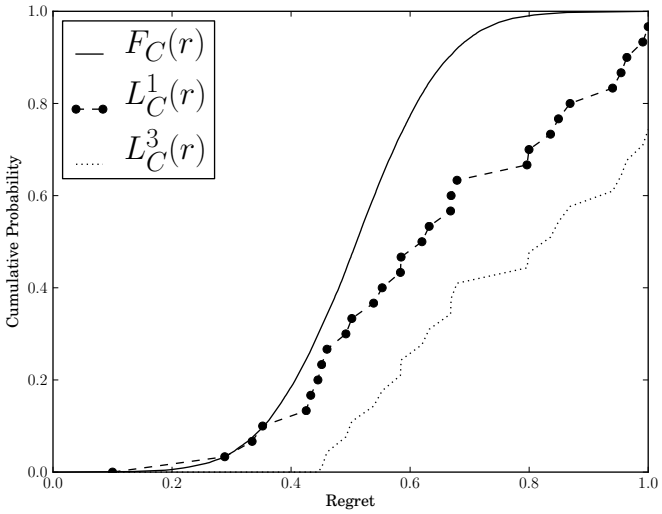


Figure 4.4: *Continuing the example from Figure 4.2. Although $L_C^1(r)$ is a lower bound of $\hat{F}_C(r)$, we do not know if $L_C^1(r)$ is a lower bound of $F_C(r)$ (since we do not know how accurately $\hat{F}_C(r)$ approximates $F_C(r)$). Therefore, we use Equation 4.16 to create another lower bound, $L_C^3(r)$, which has a high probability of being a lower bound to $F_C(r)$.*

be the overall probability of error, *i.e.* with probability p_{err},

$$PMR(d, d', u) > lPrPMR(d, d', C, p_{cd}).$$

Note when an error does occur, we can say nothing about $PR(d, d', u)$.

Finally, we adapt Equations 4.6 and 4.7 to work with lPrPMR by defining *learnt PrMR* and *learnt PrMMR* as

$$lPrMR(d, C, p) := \max_{d' \in \mathbb{D}} lPrPMR(d, d', C, p),$$

and

$$lPrMMR(d, C, p) := \min_{d \in \mathbb{D}} lPrMR(d, C, p),$$

respectively. We say $lPrMR$ is correct if

$$R(d, u) \leq lPrMR(d, C, p). \tag{4.19}$$

Lemma 7. *The lPrMR is correct with at least a probability of* $1 - p_{err}$.

Proof. Equation 4.19 is satisfied if and only if

$$\max_{d' \in \mathbb{D}} r(d, d', u) \leq \max_{d' \in \mathbb{D}} lPrPMR(d, d', C, p). \tag{4.20}$$

Let $d^a = \arg\max_d r(d, d', u)$. Then Equations 4.19 and 4.20 are both satisfied if

$$r(d, d^a, u) \leq lPrPMR(d, d^a, C, p). \tag{4.21}$$

We do not know which decision d^a is, we only know it exists. Note that Equation 4.21 is a sufficient but not necessary condition for Equations 4.19 and 4.20 to be satisfied. Since Equation 4.21 holds with probability of at least $1 - p_{err}$, so does Equation 4.19. □

4.2.3 Hypothesis-Based Regret

We now consider a different method for overcoming the prior knowledge assumption. Instead of learning about the distribution as we process each user, we start off with a set of hypotheses, $\mathbb{H}$, for what the correct distribution could be. The PrMMR calculations are now based on an aggregation of this set of hypotheses. As long as $\mathbb{H}$ contains the correct hypothesis, our calculations will be correct. Since $\mathbb{H}$ can consist of an arbitrary number of hypotheses, we can vary the size of $\mathbb{H}$ depending on how certain we are of the correct

hypothesis. The more hypotheses in $\mathbb{H}$, the fewer assumptions we need to make about the correct hypothesis. We can compensate for being relatively uncertain of the correct hypothesis by increasing the size of $\mathbb{H}$.[2] A key difference between this approach and a Bayesian one is that we do not assign any probability of correctness to the hypotheses in $\mathbb{H}$. We will show that this allows for more robust calculations.

The more hypotheses we include, the higher the calculated regret values will be. Since this could lead to a decrease in the efficiency of processing users, we provide a method that rejects incorrect hypotheses with a high probability while accidentally rejecting correct hypotheses with low probability. Since rejecting hypotheses can require additional queries for the user, we also provide a method to determine when it is beneficial to reject hypotheses. We also discuss how to create an optimal hypothesis given a set of previously-processed users. Finally, we discuss what happens if the correct hypothesis is not in $\mathbb{H}$.

A hypothesis $\mathcal{H}$ is defined by two things. First, we have the set of possible utility functions. Our hypothesis, for example. could be that all users have *exponential utility*, *i.e*

$$u_{\mathcal{H}}(x, \alpha) = 1 - e^{-\alpha/x}, \tag{4.22}$$

for some parameter α. The second thing that defines a hypothesis is a probability density function over the parameters for $u_{\mathcal{H}}$ other than x. For the utility function in Equation 4.22, an example distribution might be

$$f(\alpha) = \begin{cases} 1 & \text{if } 0 < \alpha \leq 1 \\ 0 & \text{otherwise.} \end{cases}$$

We first consider the case with only one hypothesis: $\mathbb{H} = \{\mathcal{H}\}$. We adapt our definition of $F_C(r)$ to be dependent on $\mathcal{H}$, *i.e.* $F_{\mathcal{H}|C}(r)$. This gives the cumulative distribution of the pairwise regret from choosing decision d instead of d', assuming $\mathcal{H}$ to be true and with the utility restrictions C imposed. This allows us to also adapt our definitions of PrPMR, PrMR, and PrMMR (Equations 4.5 through 4.7) to also be dependent on $\mathcal{H}$, that is,

$$hPrPMR(d, d', \mathcal{H}|C, p_{cd}) := F^{-1}_{\mathcal{H}|C}(p_{cd}), \tag{4.23}$$

$$hPrMR(d, \mathcal{H}|C, p_{cd}) := \max_{d' \in \mathbb{D}} hPrPMR(d, d', \mathcal{H}|C, p_{cd}), \tag{4.24}$$

$$hPrMMR(\mathcal{H}|C, p_{cd}) := \min_{d' \in \mathbb{D}} hPrMR(d, \mathcal{H}|C, p_{cd}). \tag{4.25}$$

[2]Since there are an uncountable number of hypotheses, we can not include all of them in $\mathbb{H}$. Therefore, using this method we need to make at least some assumptions about the correct hypothesis.

To generalize for multiple hypotheses, we must calculate Equation 4.23 for each hypothesis in $\mathbb{H}$. Then the PrPMR over multiple hypotheses is just an aggregation of these values, *i.e.*

$$hPrPMR(d, d', \mathbb{H}|C, p_{cd}) := \max_{\mathcal{H} \in \mathbb{H}} hPrPMR(d, d', \mathcal{H}|C, p_{cd}). \tag{4.26}$$

The PrMR and the PrMMR over multiple hypotheses are defined as

$$hPrMR(d, \mathbb{H}|C, p_{cd}) = \max_{d' \in \mathbb{D}} hPrPMR(d, d', \mathbb{H}|C, p_{cd})$$

and

$$hPrMMR(\mathbb{H}|C, p_{cd}) = \min_{d \in \mathbb{D}} hPrMR(d, \mathbb{H}|C, p_{cd}).$$

Definition 13. *The bound $hPrPMR(d, d', \mathbb{H}|C, p)$ is correct if*

$$r(d, d', u) \leq hPrPMR(d, d', \mathbb{H}|C, p) \tag{4.27}$$

with probability of at least p.

Proposition 3. *If $F_{\mathcal{H}|C}(r)$ is independent for all $\mathcal{H}$, then $PrPMR(d, d', \mathbb{H}|C, p)$ is correct as long as $\mathbb{H}$ contains the correct hypothesis.*

Proof. Let $\mathcal{H}^*$ be the correct hypothesis. By definition of $F_{\mathcal{H}|C}(r)$, if $F_{\mathcal{H}^*|C}(r)$ is independent of $F_{\mathcal{H}'|C}(r)$ for other hypotheses $\mathcal{H}'$, then the probability that

$$r(d, d', u) \leq PrPMR(d, d', \mathcal{H}^*|C, p), \tag{4.28}$$

is p. Since $PrPMR(d, d', \mathbb{H}|C, p)$ takes the maximum over all $PrPMR(d, d', \mathcal{H}|C, p)$, then as long as Equation 4.28 is satisfied, Equation 4.27 is also satisfied. Suppose Equation 4.28 is not satisfied. Since the probability that

$$r(d, d', u) \leq PrPMR(d, \mathcal{H}|C, p)$$

is non-negative for any hypothesis (correct or incorrect), this case cannot decrease the probability that Equation 4.27 is satisfied. □

An important implication of Proposition 3 is that while incorrect hypotheses can result in an overestimation of the $hPrPMR$, they can never affect correctness (assuming the correct hypothesis is in $\mathbb{H}$). Since we consider each hypothesis conditional with respect to the utility constraints C, $hPrMMR(\mathbb{H}|C)$ will never be larger than $MMR(C)$. For example, if $\mathcal{H}$ is that $u(x_i)$ is uniformly distributed between 0 and 1 and our set of utility constraints C says that $u(x_i)$ is between 0 and 0.5, then $\mathcal{H}|C$ says that $u(x_i)$ is uniformly distributed between 0 and 0.5.

4.2.4 Rejecting Hypotheses

With additional hypotheses our upper bound on regret, calculated using Equation 4.26, can become less accurate and more time consuming to calculate. Therefore, we need a method to reject incorrect hypotheses while minimizing the chances of accidentally rejecting the correct hypothesis.

Each time we have finished processing a user, we examine the utility constraints from that user and all previous users to see if there is any evidence against any of the hypotheses. Many hypotheses can be rejected with certainty. For example, suppose we have the hypothesis

$$\mathcal{H} : u = 2x^{r'}, \tag{4.29}$$

but the actual utilities are given by

$$u = x^r,$$

for some distributions of r and r'. If we have the outcomes $x = 9$ and $x = 16$ and $r = 0.5$, then we would have the utility values $u(9) = 3$ and $u(16) = 4$. However, there is no value for r' which satisfies these utility values and Equation 4.29. In this case, we would know, with certainty, that $\mathcal{H}$ is false.

However, there are many hypotheses that can never be rejected with certainty. For example, it is always possible that a set of utility values have been chosen uniformly at random. In these cases, the best we can do is say a hypothesis is unlikely. We need a method that can reject a hypothesis if it is unlikely enough, while minimizing the chances of accidentally rejecting the correct hypothesis; to do this, we use statistical testing.

Statistical testing compares the values we have observed while processing users against the values we would expect to see occur under a particular hypothesis. A first choice of values to compare might be the users' utility constraints against the distribution of utility values specified by a hypothesis. The difficulty with comparing utility values is that each outcome adds a new dimension to the utility space and therefore our data is multidimensional. Difficulties with testing multidimensional data include dealing with sparsity of data and possible correlations between dimensions. One possible solution, often used in Bayesian learning, is to assume independence; however, utility values are often highly correlated [44]. For example, for exponential utility values (of the form given in Equation 4.22), after the first three utility values, all other values are redundant. This suggests that assuming independence may not be a feasible approach. There are methods which are able to account for correlations when testing multidimensional data. However, to our knowledge, these methods are exponential with respect to the number of dimensions [39].

Since the goal of both probabilistic and traditional minimax regret is to minimize regret, we felt it would be more natural to compare the regret values predicted by hypotheses. Specifically, we decided to compare the observed pairwise regret between any two decisions against the pairwise regret distribution inferred by each hypothesis. In tests, we found that correlations between pairwise regret values for different decisions to be minimal. Thus, we assume that regret distributions are independent. This also helps us avoid the data sparsity problem. One problem with comparing regret values is that it is possible for different utility distributions to give the same regret values. Since we choose the decisions based on regret, and not directly on utility, if such utility distributions exist, we would not be interested in differentiating such distributions.

We can reject a hypothesis when it either overestimates or underestimates the regret. There are different, but related, methods for checking for either case. If we assume that the correct hypothesis is in $\mathbb{H}$, then underestimating regret has no effect on correctness (see Proposition 3) or on efficiency. As a result, checking for overestimation is more important.

Our method for rejecting hypotheses relies on the Kolmogorov-Smirnov (KS) one-sample test [48]. We use the KS test to compare the regret values we would see if a hypothesis was true against the regret values we see in practice. The test statistic for the KS test is

$$T^{\mathcal{H}}_{d,d',i} = \max_{r\in[0,1]} |F_{d,d',i,\mathcal{H}}(r) - \hat{F}_{d,d',i}(r)|, \tag{4.30}$$

where $\hat{F}(r)$ is the empirical distribution function defined in Equation 4.9. For notational brevity, we refer only to $T^{\mathcal{H}}$ instead of $T^{\mathcal{H}}_{d,d',i}$. An example of finding the KS statistic is shown in Figure 4.5.

If $\mathcal{H}$ is correct, then as i approaches infinity,

$$\sqrt{i} \cdot T$$

converges to the *Kolmogorov distribution* which does not depend on $F_{\mathcal{H}}$. We reject $\mathcal{H}$ if

$$\sqrt{i} \cdot T \geq K_\alpha, \tag{4.31}$$

where K_α is such that

$$\Pr(K \leq K_\alpha) = 1 - \alpha$$

and K is the cumulative distribution function of the Kolmogorov distribution.

The statistic T measures the maximum absolute difference between $F_{\mathcal{H}}(r)$ and $\hat{F}(r)$. We can break T up into two parts, T_{over} and T_{under}. Let

$$T_{over} := \max\{0, \max_r(\hat{F}(r) - F_{\mathcal{H}}(r))\}, \tag{4.32}$$

be the evidence against $\mathcal{H}$ as the result of $\mathcal{H}$ overestimating the regret. As shown in Figure 4.1, if $F_{\mathcal{H}}(r)$ is overestimating the regret, then $F_{\mathcal{H}}(r) < \hat{F}(r)$. Similarly, let

$$T_{under} := \max\{0, \max_{r}(F_{\mathcal{H}}(r) - \hat{F}(r))\}, \tag{4.33}$$

be the evidence against $\mathcal{H}$ as the result of $\mathcal{H}$ underestimating the regret. As a result, $T = \max\{T_{over}, T_{under}\}$.

Rejecting a Hypothesis when it Overestimates the Regret

Since we do not know $\hat{F}(r)$, we rely on the lower bound, $L^1(r)$, given in Equation 4.12 and assume equality to cover the worst case. In this case, we can give a lower bound to Equation 4.32 with

$$T_{over} \geq \max\{0, \max_{r}(L^1(r) - F_{\mathcal{H}}(r))\}, \tag{4.34}$$

This statistic is illustrated in Figure 4.5.

If $\mathcal{H}$ is true, then the probability that we incorrectly reject $\mathcal{H}$ based on $T^{\mathcal{H}}_{d,d',i}$ for a specific set of decisions $\{d, d'\}$ is at most α. However, since we examine $T^{\mathcal{H}}_{d,d',i}$ for every possible combination of decisions, the probability of incorrectly rejecting $\mathcal{H}$ is much higher. (This is known as the *multiple testing problem*.) Our solution is to use the *Bonferroni Method* where we reject $\mathcal{H}$ if

$$\max_{\{d,d'\} \subseteq D} \sqrt{i} \cdot T^{\mathcal{H}}_{d,d',i} \geq K'_{\alpha},$$

where [69]

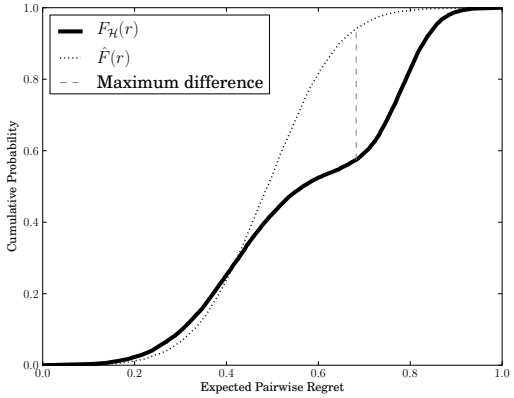
$$\Pr(K \leq K'_{\alpha}) = \frac{1-\alpha}{|D|(|D|-1)}. \tag{4.35}$$

Using this method, the probability of incorrectly rejecting $\mathcal{H}$ is at most α.

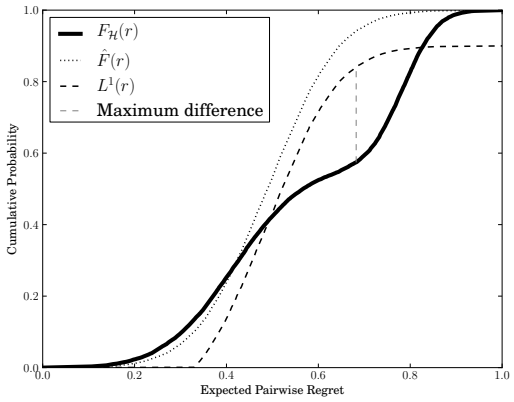
Rejecting a Hypothesis when it Underestimates the Regret

Our next goal is to measure T_{under}, given in Equation 4.33, which measures the evidence against $\mathcal{H}$ as the result of $\mathcal{H}$ underestimating the regret. However, since regret is private, we cannot compute $\hat{F}(r)$ and therefore, we also cannot compute T_{under}. Instead, we provide a lower bound on T_{under} in a similar approach as we used with bounding T_{over}. The basic set up is shown in Figure 4.6.

Since we are concerned with the case where $F_{\mathcal{H}}(r)$ is above $\hat{F}(r)$ (as shown in Figure 4.6), to bound T_{under} we need to find an upper bound on $\hat{F}(r)$. To do so, we need to find a lower bound on $r(d, d', u)$ given $PMR(C)$. We rely on the following proposition.

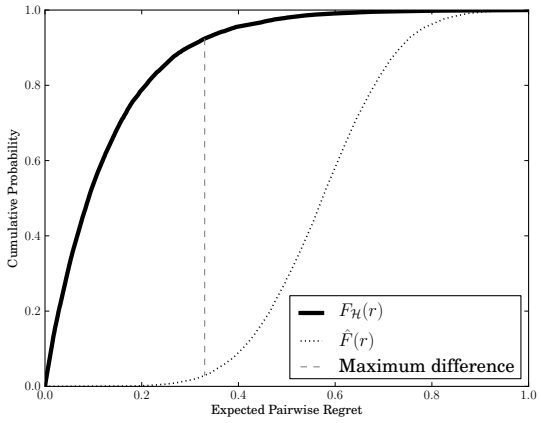


(a) KS test comparing $F_{\mathcal{H}}(r)$ against $\hat{F}(r)$.

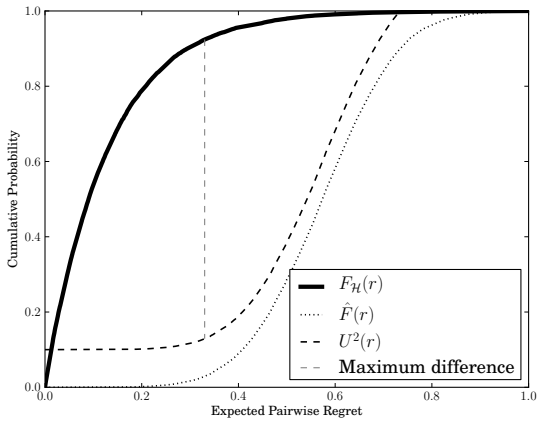


(b) KS test comparing $F_{\mathcal{H}}(r)$ against $L^1(r)$.

Figure 4.5: *An example of the Kolmogorov-Smirnov one-sample test. Our goal is to find evidence against the hypothesis $\mathcal{H}$. Ideally, we want to do this by comparing the regret we would expect to see if $\mathcal{H}$ was true ($F_{\mathcal{H}}(r)$, Equation 4.23) against the regret we have actually seen ($\hat{F}(r)$, Equation 4.9). This situation is shown in Figure 4.5(a). The KS test focuses on the maximum absolute difference between the two distributions, i.e. Equation 4.30. However, since the actual regret is private, we must use $L^1(r)$ from Equation 4.12 as a lower bound for Equation 4.9. This situation is shown in Figure 4.5(b). Here the KS test focuses on the maximum difference as defined in Equation 4.34. Note how the maximum difference in Figure 4.5(b) is a lower bound for the maximum difference in 4.5(a).*



(a) KS test comparing $F_{\mathcal{H}}(r)$ against $\hat{F}(r)$.



(b) KS test comparing $F_{\mathcal{H}}(r)$ against $U^2(r)$.

Figure 4.6: *An example of using the KS test to reject a hypothesis which underestimated the regret. As with the example in Figure 4.5, we want to compare the regret predicted by the hypothesis $\mathcal{H}$, shown by the function $F_{\mathcal{H}}(r)$, against the actual regret seen, shown by the function $\hat{F}(r)$. Since regret is private, we cannot calculate $\hat{F}(r)$. If we compared $F_{\mathcal{H}}(r)$ against $L^1(r)$, a lower bound of $\hat{F}(r)$, as we did in Figure 4.5, we would find zero evidence against $F_{\mathcal{H}}(r)$. Instead, we compare $F_{\mathcal{H}}(r)$ against $U^2(r)$, an upper bound of $\hat{F}(r)$, as shown in Figure 4.6(b).*

Proposition 4. *Given the utility constraint set C, for the pair of decisions d and d' let*

$$PMR_{err} = \sum_{x \in \mathbb{X}} \left| \Pr_{d'}(x) - \Pr_{d}(x) \right| \left[C_{\max}(x) - C_{\min}(x) \right]$$

then the difference between the pairwise maximum regret and PMR_{err} is bounded by

$$PMR(d, d', C) - PMR_{err} \leq r(d, d', u). \tag{4.36}$$

Proof. We need to show that

$$PMR(d, d', C) - r(d, d', u) \leq PMR_{err}.$$

If we let

$$u^a_{d,d'} = \arg\max_{u' \in C} r(d, d', u'),$$

then

$$\begin{aligned} PMR(d, d', C) - r(d, d', u) &= \sum_{x \in \mathbb{X}} \left[\Pr_{d'}(x) - \Pr_{d}(x) \right] u^a_{d,d'}(x_i) - \sum_{x \in \mathbb{X}} \left[\Pr_{d'}(x) - \Pr_{d}(x) \right] u(x_i) \\ &= \sum_{x \in \mathbb{X}} \left[\Pr_{d'}(x) - \Pr_{d}(x) \right] \left[u^a_{d,d'}(x_i) - u(x_i) \right] \end{aligned} \tag{4.37}$$

We note that if $\Pr_{d'}(x_i) \geq \Pr_d(x_i)$, to maximize regret, we need to maximize $u^a_{d,d'}(x_i)$ and therefore $u^a_{d,d'}(x_i) - u(x_i) \geq 0$. If $\Pr_{d'}(x_i) < \Pr_d(x_i)$, to maximize regret, we need to minimize $u^a_{d,d'}(x_i)$ and therefore $u^a_{d,d'}(x_i) - u(x_i) \leq 0$. As a result, we can rewrite Equation 4.37 as

$$\begin{aligned} &\sum_{x \in \mathbb{X}} \begin{cases} \left[\Pr_{d'}(x) - \Pr_d(x) \right] \left[u^a_{d,d'}(x_i) - u(x_i) \right] & \text{if } \Pr_{d'}(x) \geq \Pr_d(x) \\ - \left[\Pr_{d'}(x) - \Pr_d(x) \right] \left| u^a_{d,d'}(x_i) - u(x_i) \right| & \text{otherwise} \end{cases} \\ =&\sum_{x \in \mathbb{X}} \begin{cases} \left[\Pr_{d'}(x) - \Pr_d(x) \right] \left[u^a_{d,d'}(x_i) - u(x_i) \right] & \text{if } \Pr_{d'}(x) \geq \Pr_d(x) \\ \left| \Pr_{d'}(x) - \Pr_d(x) \right| \left| u^a_{d,d'}(x_i) - u(x_i) \right| & \text{otherwise} \end{cases} \\ =&\sum_{x \in \mathbb{X}} \left| \Pr_{d'}(x) - \Pr_{d}(x) \right| \left| u^a_{d,d'}(x_i) - u(x_i) \right| \end{aligned} \tag{4.38}$$

We maximize $\left| u^a_{d,d'}(x_i) - u(x_i) \right|$ when $u^a_{d,d'}(x_i) = C_{\max}(x_i)$ and $u(x_i) = C_{\min}(x_i)$. As a result,

$$|u^a_{d,d'}(x_i) - u(x_i)| \leq C_{\max}(x_i) - C_{\min}(x_i).$$

Therefore, we can bound Equation 4.38 by

$$\leq \sum_{x \in \mathbb{X}} \left| \Pr_{d'}(x) - \Pr_{d}(x) \right| [C_{\max}(x) - C_{\min}(x)]$$

□

Since we now have a lower bound on $r(d, d', u)$, we can give an upper bound to $\hat{F}(r)$ similar to the lower bound we gave in Equation 4.12,

$$\hat{F}(r) \leq U^2(r) := \frac{1}{i-1} \sum_{j<i} I(PMR(d, d', C) - PMR_{err} \leq r). \tag{4.39}$$

As a result, we can give the following lower bound for T

$$T_{under} \geq \max\{0, \max_{r}(F_{\mathcal{H}}(r) - U^2(r))\}. \tag{4.40}$$

We reject $\mathcal{H}$ is either T_{under} or T_{over} exceeds the bound in Equation 4.31.

Note that the inequality in Equation 4.36 is a tight constraint. Specifically, if we choose the set of utility values

$$u(x) = \begin{cases} C_{\min}(x) & \text{if } \Pr_{d'}(x) > \Pr_{d}(x) \\ C_{\max} & \text{otherwise,} \end{cases} \tag{4.41}$$

we have equality in Equation 4.36.

Choosing the Best Hypothesis

The flexibility in choosing how many hypotheses are in $\mathbb{H}$ means that we do not have to make strong assumptions about what the correct hypothesis is. However, even for a relatively restricted set of hypotheses, listing each hypothesis may be too time consuming and ineffective. For example, suppose we believe that users all have a utility function of the form

$$u(x) = x^{\beta}$$

where β is chosen uniformly at random on the closed interval $[lb, ub]$. Despite this restricted set of possible utility functions, there are many hypotheses that are consistent with it. We can create a new hypothesis by changing either lb, ub or both. Even if we discretize the

possible values, there may still be too many hypotheses to feasibly deal with. For example, if we considered values only in increments of 0.1, we would have the following sets of parameters for (lb, ub):

$$\begin{array}{ccccc} (0, 0.1) & (0, 0.2) & \dots & (0, 1) \\ & (0.1, 0.2) & \dots & (0.1, 1) \\ & & \ddots & \vdots \\ & & & (0.9, 1) \end{array}$$

The sum of all of these possible sets of parameters would give us 55 possible hypotheses.

One possible way to improve efficiency is to specify a set of possible hypotheses dependent on some common set of parameters. In the previous example, this set of parameters would be lb and ub. If we defined a metric for evaluating how good each hypothesis is, we could choose the hypothesis with the best parameters. As a result, we could then just specify the set of possible hypotheses instead of listing each individual one.

This problem setting is very similar to the method of Bayesian *maximum a posteriori probability* (MAP) estimation [44]. The key difference is that MAP estimation tries to choose the most likely values. However, our methods do not find evidence in favour of any hypothesis, only evidence against. Lack of evidence against a hypothesis is not necessarily evidence in favour. In statistical testing, the ability to find evidence in favour of a hypothesis is known as *statistical power* [69]. The power of a statistical test is the probability of correctly rejecting an incorrect hypothesis: if the power of a test is 1, then a statistical test will always reject an incorrect hypothesis. Hence, if the power of a test is 1, or close to 1, we may take lack of evidence against as evidence in favour.

In the limit, as the sample size increases, the power of the KS test approaches 1: however, our method is only an approximation of the KS test since we do not know the exact regret values for each user [48]. Furthermore, users do not care about the eventual accuracy of our method, they only care how accurate the process is at the time when they are being processed. The power of our method for any specific user depends on both the number of users we have already processed and the accuracy of the utility constraints for each of those users.

We avoid these complications by focusing on the evidence against a hypothesis. As a result, we define an optimal hypothesis as the one with the least evidence against it, or the *least-likely incorrect hypothesis* (LLIH). In the two previous sections, we provided methods for giving lower bounds on the amount of evidence against a hypothesis by giving a lower bound on T. It may also be useful to find an upper bound on T. For example, suppose we have two hypotheses $\mathcal{H}_1$ and $\mathcal{H}_2$. If all we know is that $T^{\mathcal{H}_1} > 1$ and $T^{\mathcal{H}_2} > 1.1$ we would choose $\mathcal{H}_1$ as being the hypothesis with the least amount of evidence against it. However,

if we also knew that $T^{\mathcal{H}_1} < 3$ and $T^{\mathcal{H}_2} < 1.5$, that is, in the worst case, there is more evidence against $\mathcal{H}_1$, we might choose $\mathcal{H}_2$ as our optimal hypothesis.

To find an upper bound on T, we calculate upper bounds on both T_{over} and T_{under}. These bounds easily follow from our definitions of T_{over} and T_{under}, *i.e.*

$$\begin{aligned} T_{over} &= \max\{0, \max_r\{\hat{F}(r) - F_{\mathcal{H}}\}\}, \\ &\leq \max\{0, \max_r\{U^2(r) - F_{\mathcal{H}}\}\}, \end{aligned} \tag{4.42}$$

and

$$\begin{aligned} T_{under} &= \max\{0, \max_r\{F_{\mathcal{H}} - \hat{F}(r)\}\} \\ &\leq \max\{0, \max_r\{F_{\mathcal{H}} - L^1(r)\}\}. \end{aligned} \tag{4.43}$$

Choosing the LLIH works as follows. Given a functional form for the utility values and a range of parameters, we search through the parameter space looking for the parameter values with the least evidence against them as measured by T. Since we only have a range for what T could be, we have some flexibility in how we measure the evidence against a particular set of parameter values. For example, we could choose the set of parameters that minimize the maximum possible value for T. If the range of values for T is too large, the maximum T value may be too pessimistic. We could instead take an optimistic approach and choose the parameter values which minimize the minimum possible value for T. Finally, we could decide based on some combination of minimum and maximum possible values, *e.g.* the average. If there is strong evidence against even the least likely incorrect hypothesis, this is evidence against that particular function form of the utility values.

The accuracy of the range of T is heavily dependent on the accuracy of the utility constraints. For example, if we knew the exact utility values for each user, we would know the exact value of T. Having a small range for T allows us to choose better parameter values, which can allow us to process future users with fewer queries. Since the accuracy of the utility constraints is dependent on the number of queries we ask each user, there is a tradeoff between processing the current users quickly and improving efficiency for the long term.

One way to balance these goals is to treat the initial set of users as a focus group. For these users, the main objective would be to maximize the accuracy of their utility constraints. For example, suppose we had a maximum of 30 queries for each user. If we reached the desired regret after only 10 queries, we would use the remaining 20 queries to

maximize the accuracy of the utility constraints, where accuracy is inversely proportional to the size of the largest utility gap. Only once we have processed each of the users in the focus group would we find the LLIH. These users would be the equivalent of training data for Bayesian reasoning. As we process subsequent users, we would keep on checking to see if there was evidence against the original LLIH. If we did find substanial evidence against it, we would need to find a new LLIH.

What if the Correct Hypothesis is not in $\mathbb{H}$?

There is no guarantee that the correct hypothesis will always be in $\mathbb{H}$. If the correct hypothesis is not in $\mathbb{H}$, our proof of correctness does not hold. If we have low confidence that the correct hypothesis is in $\mathbb{H}$, there are several options. The first step is to add more hypotheses to $\mathbb{H}$. Using the Least LLIH approach, we only need to add one hypothesis for each type of utility function, *e.g.* exponential, linear, *etc.* We would then calculate the *least-likley incorrect* (LLI) exponential hypothesis, the LLI linear hypothesis, *etc.* However, if we are unsure of the basic form of the utility function, it is possible that we cannot add enough hypotheses to $\mathbb{H}$ to ever be confident that the correct hypothesis is in $\mathbb{H}$. If there is strong evidence against each of the LLI hypotheses, this strong evidence that we do not know what the utility function type is.

This is the same problem that statistical regression and least squares fitting has - if we are trying to fit data to the wrong functional form, it may be impossible to get a good fit. If we cannot get a good fit, we can either fall back on minimax regret or probabilistic regret without hypotheses. We can also treat an initial group of users as a focus group and use their preferences to try to discover what other possible utility functions we should consider.

4.2.5 A Probably Approximately Correct Approach to Probabilistic Regret

Probably Approximatley Correct (PAC) learning is an area of AI research [46, 60]. The goal of PAC learning is to learn a concept based on a set of labelled examples. For example, the concept could be real numbers greater than 10 [46]. The actual concept is never revealed. Instead, a set of random numbers is generated and labelled 1 if they satisfy the concept, *e.g.* that number is greater than 10, and 0 otherwise. Based on these examples, PAC learning aims to create a hypothesis which can correctly label any real number with a high probability. A key goal in PAC learning is to create the hypothesis in polynomial time.

Non-parametric statistics and PAC learning are closely related [62]. A key difference in our work is that we are not interested in whether either lPrMMR or hPrMMR can work in polynomial. This is a definite area for future research. While many of the results in PAC learning are built on results from non-parametric statistics, we have not found any reference of the Dvoretzky-Kiefer-Wolfowitz inequality in PAC literature. Understanding how the Dvoretzky-Kiefer-Wolfowitz inequality could be viewed from a PAC perspective would allow us to frame our work in a PAC setting. This is another area for future research.

4.2.6 Other Uses of Non-Parametric Statistics in AI

The main statistical tools we used in this chapter; empirical distribution functions, the Dvoretzky-Kiefer-Wolfowitz inequality and the Kolmogorov-Smirnov one-sample test have a long history of use in non-parametric statistics. We make no claim that the idea of using these tools to analyze data is novel in any way. Even in AI, there has been considerable work done that has used these tools. For example, researchers have used non-parametric statistics to examine empirical auction data and compare the results to theoretical predictions [28].

4.3 How to Optimize Regret Calculations

In this section we examine methods for optimizing the performance of both lPrMMR and hPrMMR.

4.3.1 Optimizing lPrMMR

Our first method, *optimal parameter selection*, allows us to select the optimal parameter values for lPrMMR. The second method, *weighted start heuristic*, explores the tradeoff between additional querying for some users and using the resulting additional information to improve efficiency for other users.

Optimal Parameter Selection

As shown in Equation 4.18, our overall probability of error for lPrMMR is controlled by two parameters, α and p_{cd}. Thus, for a fixed value for p_{err} we may view p_{cd} as a function

of α, *i.e.*,

$$p_{cd} = \frac{1 - p_{err}}{1 - \alpha}. \tag{4.44}$$

It may be possible that some values of α are able to achieve a lower lPrMMR than other values. Since there are no requirements that α or p_{cd} be fixed in advance of the elicitation process, we are free to vary these values throughout the process. As a result, each time we calculate the PrMMR, the *optimal parameter selection* (OPS) method searches through all values of α to find the value which minimizes the lPrMMR. For the sake of simplicity, we discretize the set of possible values of α.

Weighted Start Heuristics

Since the constraints for one user are used to help process future users, for each user we face a tradeoff between minimizing queries and refining the constraint set for future benefit. It may be worthwhile to do additional querying of some users, especially the initial users. *Weighted start heuristics* (WSHs) are a set of heuristics which include any method of giving the first i users additional queries.

We propose three WSHs. The first WSH is to set a minimum number of queries. For example, we could set a minimum of 20 queries, then we would always query the first i users at least 20 times, even if we have reached the given regret threshold already. The second possibility is to require a lower level of regret for the first i users. For example, for the first i users we could have a regret threshold of 0.005 while for the subsequent users we would only have a regret threshold of 0.01. It will generally take additional queries to reach this lower threshold. The third WSH is to require a lower probability of error, as given by Equation 4.18. Similar to the previous method, it will generally require more queries to reach a lower probability of error.

4.3.2 Optimizing hPrMMR

We next consider a heuristic for helping us to effectively reject heuristics when using hPrMMR.

Heuristics for Rejecting Hypotheses

Since incorrect hypotheses can result in higher regret estimates, we would like to reject the incorrect hypotheses as quickly as possible. A major factor in how quickly we can

reject incorrect hypotheses is how accurate the utility constraints are for the users we have processed. In many cases, it may be beneficial in the long run to spend some extra time querying the initial users for improved utility constraints. To study these tradeoffs between short-term and long-term efficiency we used a simple heuristic, $Reject(j)$. With the $Reject(j)$ heuristic, we initially query every user for the maximum number of queries. Once we have rejected j hypotheses, we query only until the PrMMR is below the given threshold. We can vary j based on the number of hypotheses we want to reject. While this means that the initial users will be processed inefficiently, we will be able to quickly reject incorrect hypotheses and improve the long term efficiency over the population of the users.

4.4 Experimental Results

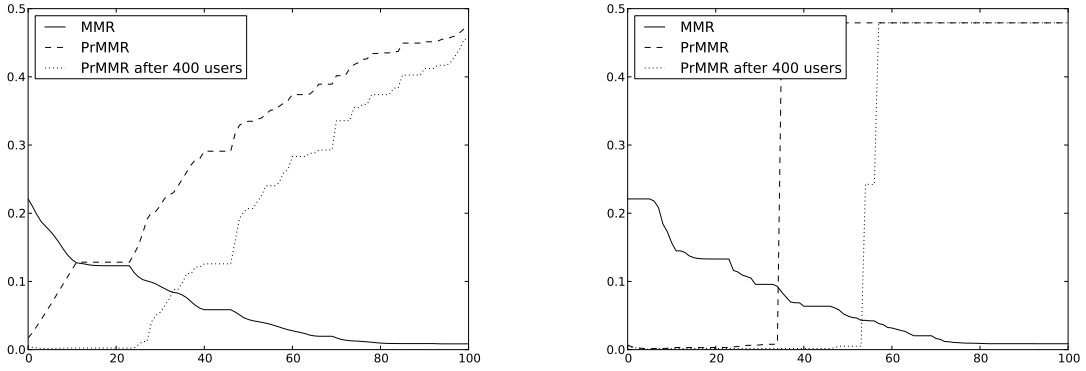
In this section, we present our experimental results. The section is divided into two: we present the results for both learnt probabilistic regret (Section 4.4.1) and probabilistic regret with hypotheses (Section 4.4.2). For all of these experiments we assumed that users' preferences followed the axioms of EUT. Combining the work in this chapter with that in the previous chapter is purely a technical exercise.

4.4.1 Learnt-Probabilistic Regret Results

Setup

For our experiments, we investigated a simplified market for buying electricity on the Smart Grid. In this market, each day people pay a lump sum of money for the next day's electricity. We assume one aggregate utility company that decides on a constant per-unit price for electricity which determines how much electricity each person receives. We assume a competitive market where there is no profit from speculating.

A person's decision, c, is how much money to pay in advance. For simplicity, we consider only a finite number of possible amounts. There is uncertainty both in terms of how much other people are willing to pay and how much capacity the system will have the next day. However, based on historical data, we can estimate, for a given amount of payment, the probability distribution for the resulting amount of electricity. Again, for simplicity, we consider only a finite number of outcomes. Our goal is to process a set of Smart Grid users and help them each decide on their optimal decision.



(a) Mean Regret

(b) Median Regret

Figure 4.7: *The average regret over 100 queries using either MMR or lPrMMR with 500 simulated users. Figure 4.7(a) shows the mean regret and Figure 4.7(b) shows the median regret. We also show the performance of lPrMMR on the last 100 users. Note that we are always using the maximum number of queries, regardless of when we reach the given threshold regret. As a result, these experiments compare the basic behaviour of lPrMMR and MMR, not their relative efficiency.*

Each person's overall utility function is given by

$$u(c, x) = u_{\text{elect}}(x) - c,$$

where x is the amount of electricity they receive. For each user, we used

$$u_{\text{elect}}(x) = x^{\beta}, \tag{4.45}$$

where $0 \leq \beta \leq 1$ is chosen uniformly at random for each user. We are interested in utility functions of the form given in Equation 4.45, since it is often used to describe peoples' preferences in experimental settings [59].

Results

We began by studying the basic behaviour of probabilistic minimax regret compared to (non-probabilistic) minimax regret. Figure 4.7 shows the average (both mean and median)

regret after each query using either MMR or lPrMMR after processing 500 simulated users. For both types of regret, we used the HLG heuristic to select queries - this means that for each user, we asked the same set of queries regardless of the type of regret. For lPrMMR set $\alpha = 0.3$ and $p_{cd} = 0.7$ which means that $p_{err} = 0.51$, so lPrMMR is basically selecting the median regret. We did not use any optimization or heuristics for lPrMMR. For this initial set of experiments, we are interested in comparing the basic behaviour of the two types of regret, instead of their relative efficiency. As a result, we always used the maximum number of queries, regardless of when the threshold regret was reached.

The minimax regret decreases monotonically over time and the mean and median values are roughly the same. The probabilistic minimax regret initially decreases, then remains steady for a time and then increases. This is explained by a tradeoff between the accuracy of the utility constraints and the number of compatible users. This tradeoff occurs in calculating the lower bound in Equation 4.16. With each additional query, we refine the utility constraints for the current user. This improves the accuracy of our empirical distribution function $\hat{F}_C(r)$ which also improves the accuracy of our lower bound function in Equation 4.16. At the same time, with each additional constraint, we have fewer compatible users. This results in an increase in our error value ϵ from Equation 4.14 which decreases the accuracy of our lower bound function. As we refine our utility constraints, the improved accuracy of $L^1_C(r)$ is the dominant term in Equation 4.16 and the probabilistic regret decreases. However, eventually we have so few compatible users that the ϵ value becomes the dominant term and the lPrMMR increases. As we process more users, the set of compatible users increases. For later users, the initial lPrMMR values should be lower and it should take more queries before the probabilistic regret increases. In Figure 4.7, we have also plotted the average probabilistic regret for the last 100 users. We see that the probabilistic regret is initially lower for the last 100 users and for those users it takes more queries before the probabilistic regret starts to increase.

Figure 4.7 also shows the median regret values. The mean and median minimax regret are roughly the same. However, we see that the increase in the median lPrMMR is more immediate than the increase in the mean lPrMMR. This shows that for individual users when the probabilistic regret starts to increase the increase is immediate and not gradual.

Finally, Figure 4.7 shows that eventually lPrMMR becomes larger than the minimax regret. For these experiments, this does not reflect on the efficiency of lPrMMR. Instead, this is a result of the combination of always using the maximum number of queries and the fact, as previously mentioned, that the lPrMMR will always eventually increase. If we were querying only until the given regret threshold was reached, we would see a different result. We explore this setting in subsequent simulations. Since it is possible for the lPrMMR to

Regret	Mean,Median	% not solved	% error
Minimax	75.09, 76	0	N.A.
$lPrMMR$ ($\alpha = 0.3$, $p_{cd} = 0.7$)	49.11, 50	0	0

(a) Without Monotonicity Constraint

Regret	Mean,Median	% not solved	% error
Minimax	34.95, 42	0	N.A.
$lPrMMR$ ($\alpha = 0.3$, $p_{cd} = 0.7$)	26.31, 26	0	0

(b) With Monotonicity Constraint

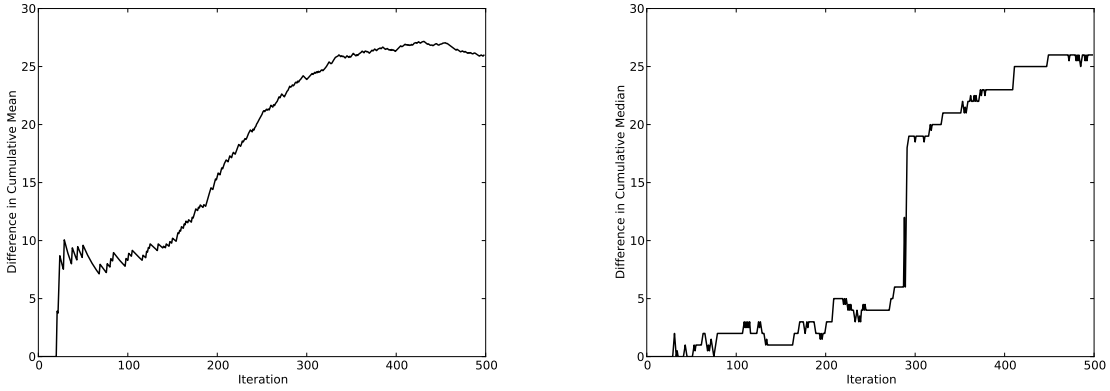
Table 4.2: *Mean and median number of queries needed per user to achieve a regret of at most 0.01 using the HLG query selection heuristic. Results are given for both MMR and lPrMMR with and without the monotonicity constraint. Percentage of users not solved, i.e. regret of at least 0.01 after 20 queries, is also given. For the probabilistic results, the percentage of users resulting in an error is also given, i.e. the percentage for which the probabilistic minimax regret is less than the actual regret.*

become larger than the MMR, for the rest of the experiments, we let

$$lPrMMR(C) = \min\{lPrMMR(C), MMR(C)\}. \tag{4.46}$$

With this restriction, using lPrMMR will never give worse results than MMR.

We compared the basic performance of MMR and PrMMR. In Table 4.2, we show the mean and median number of queries needed to reach a maximum regret of 0.01. For both types of regret we used the HLG elicitation heuristic from Section 2.2.3. For PrMMR set $\alpha = 0.3$ and $p_{cd} = 0.7$ which means that $p_{err} = 0.51$, so PrMMR is selecting the median regret. We did not use any optimization or heuristics for PrMMR. Without the monotonicity constraint, the mean number of queries using MMR is 75 and using PrMMR is 49. With the monotonicity constraint, the mean number of queries using MMR is 35 and using PrMMR is 26. In both cases, PrMMR easily out performs MMR. Without the monotonicity constraint PrMMR is 35% faster than MMR. However, with the monotonicity



(a) Mean Difference

(b) Median Difference

Figure 4.8: *A comparison of the relative performance of lPrMMR and MMR using Equation 4.47. Figure 4.8(a) shows the difference based on mean number of queries and Figure 4.8(b) shows the difference based on median number of queries. The lPrMMR values were based on setting* $\alpha = 0.3$ *and* $p_{cd} = 0.7$ *and without any optimizations.*

constraint, PrMMR is only 25% faster. This suggests that the more we know about the users' utility values, the less help PrMMR is. The fact there was never an error when using PrMMR shows just how much MMR can overestimate the actual regret.

For the initial users, the number of compatible users will be so small that the error bound given in Equation 4.14 will result in only trivial values for lPrMMR. Thus, while the performance of lPrMMR will never be worse than MMR (because of Equation 4.46), for the initial set of users there will be no difference in the performance. We were interested in finding the smallest number of users to process which would result in lPrMMR noticeably outperforming MMR. Let Q_i^P be the number of queries needed to process user i using lPrMMR and let Q_i^M be similarly defined using MMR. After processing user i, we can measure the relative cumulative performance of lPrMMR and MMR up to user i by looking at the difference

$$diff(i) := average\{Q_0^M, \ldots, Q_i^M\} - average\{Q_0^P, \ldots, Q_i^P\}, \tag{4.47}$$

where the average is either the mean or median. Figure 4.8 shows the values for Equation 4.47 as we process the 500 users. The PrMMR values used to create Figure 4.47 were based on setting $\alpha = 0.3$ and $p_{cd} = 0.7$ and without any optimizations.

Regret	Mean/Median	% not solved	% error
$p_{err} = 0.5$	42.68 34	0	0
$p_{err} = 0.3$	61.6 73	0	0
$p_{err} = 0.1$	75 76	0	0

Table 4.3: *Mean and median number of queries to process users using lPrMMR using the optimal parameter selection method with a varying probability of error.*

Since users are chosen i.i.d., any difference might be the result of random chance. Instead, we look for when the difference in Equation 4.47 becomes statistically significant. Results are shown for both mean and median differences. To determine statistical significance, we rely on the test statistic

$$W_i := \frac{diff(i)}{\hat{se}_i} \tag{4.48}$$

where

$$\hat{se}_i = \sqrt{\frac{s^2_{P,i} + s^2_{M,i}}{2}} \tag{4.49}$$

with $S^2_{P,i}$ the sample variance of $\{Q^P_0, \ldots, Q^P_i\}$ and $S^2_{M,i}$ similarly defined. We reject W_i if $|W| > z_{\alpha/2}$ where $z_{\alpha/2}$ is the value of the probability distribution function for the Gaussian function at $\alpha/2$. If we reject W_i, then the probability of the difference after i being the result of random chance is at most α. For our experiments, we used a value of $\alpha = 0.01$. For our experiments, Equation 4.47 became statistically significant after 22 users when using the mean and 32 users when using the median.

The results in Figure 4.8 show that the mean difference initially increases much faster than the median difference. In fact, for almost 300 users the median difference remains below 5 queries while the mean difference increases to over 20. This means that for at least 50% of all initial users, the difference between PrMMR and MMR is modest. However, with users for whom PrMMR makes a difference of more than 5 queries, the difference is substantial. After 300 users, the median difference increases significantly and in the end the mean and median differences are roughly equivalent.

We next examined how well optimal parameter selection (OPS) works. Table 4.3 shows the mean number of queries when using PrMMR and OPS for different values of p_{err}. With $p_{err} = 0.5$, the mean and median number of queries improves to 42 and 34 respectively. We can compare this to the values in Table 4.2. While there is only a moderate improvement in

Probability of Error	Mean/Median for all users	Mean/Median for first 400 users	Mean/Median for last 100 users
0.5	42.7, 34	42.9, 31	42.1, 36
0.3	61.6, 73	66.8, 74	40.9, 24
0.1	71.1, 76	75.5, 76	73.2, 74

Table 4.4: *The results from Table 4.3 divided between the first 400 users and the last 100 users.*

Probability of Error	Mean/Median for all users	Mean/Median for first 400 users	Mean/Median for last 100 users
0.5	59.7, 76	79.3, 76	0.79, 0
0.3	65.2, 76	79.8, 76	6.5, 3
0.1	77.8, 76	79.9, 76	72.7, 73.5

Table 4.5: *Mean and median number of queries needed to process users with lPrMMR+OPS+WSH. The probability of error is varied. With WSH, we require a minimum of 75 queries for the first 400 users.*

the mean number of queries, from 49 to 42, the median number of queries improves from 50 to 34. This shows that for a majority of users, OPS is able to make a considerable difference but there are some users for whom OPS makes no difference. The median number of queries with OPS is close, though not equal, to the median number of queries using PrMMR with no optimization but assuming monotonicity. This suggests that OPS may be roughly as powerful as assuming monotonicity.

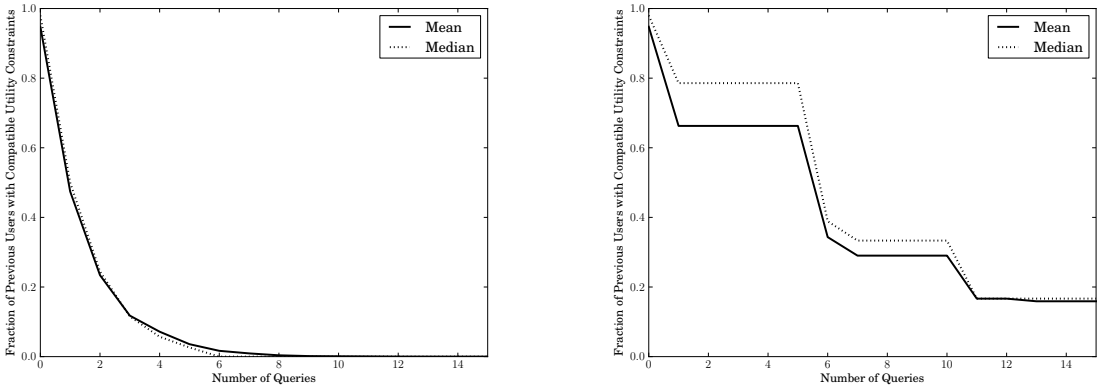
Table 4.3 also shows the results for setting $p_{err} = 0.3$ and 0.1. In the first case, the results are roughly half way in between MMR and PrMMR. With $p_{err} = 0.1$ the results are identical to using MMR. However, for all of these results, we still do not have any error occurring. This suggests that we can improve on these results. If we had reached the optimal performance of PrMMR then with $P_{err} = 0.1$ we would expect to see errors occurring 10% of the time. We can improve on these results either by having more accurate utility constraints or by processing more users.

To obtain more accurate constraints, we experimented with the weighted start heuristic. The weighted start heuristic is based on the idea of requiring a minimum number of queries from initial users. Therefore, we need to measure the success of the weighted start heuristic by three things: what the overall performance is, what the performance is for the initial users and what the performance is for the end users.

As a benchmark, we first took the results for PrMMR+OPS from Table 4.3 and found the average number of queries for the first 400 users as well as the last 100 users. The results are shown in Table 4.4. These results illustrate a tradeoff in information requirements. On the one hand, with a high probability of error, we do not need highly accurate utility constraints to process each user, *i.e.* for each user, we have a low information requirement. However, this also means that we are obtaining little information from each user to help with processing future users. This can be seen with $p_{err} = 0.5$ where the average number of queries remains relatively constant for all 500 users. At the other extreme with $p_{err} = 0.1$, we need more accurate utility constraints and hence we are obtaining more information from each user. However, with the decrease in the probability of error, we also need more information to process each user: with $p_{err} = 0.1$, despite the fact that we are obtaining more information from each user, the amount of information we need per user has increased even more. This is shown by the fact that with $p_{err} = 0.1$, the average number of queries again remains constant. With $p_{err} = 0.3$, we have found a "sweet spot" where we are actually gaining enough information from each user to improve the efficiency with which we process future users. As a result, the average number of queries decreases as we process additional users.

Our weighed start heuristic (WSH) experiments involved requiring a minimum of 75 queries for the first 400 users. The results are shown in Table 4.5 and can be compared directly against the results in Table 4.4. For a probability of error equal to 0.5, we can process the last 100 users with an average of less than one query each. This is a substantial improvement over the average for the first 400 users and the last 100 users in Table 4.4. However, the price we pay is that the average number of queries for the first 400 users increases noticeably. For a probability of error equal to 0.3, we again see a substantial improvement in the average number of queries needed to process the last 100 users. Note that with $p_{err} = 0.3$, the median number of queries for the first 400 users increases by only 2. This means that most of the initial 400 users are affected only minimally by the minimum query requirement. The mean number of queries for the first 400 users increases by 13 queries. The difference between the increase in the mean verses median number of queries means that while most users are not affected by the increase in queries, those users who are affected require a substantial number of additional queries. With $p_{err} = 0.1$, the average number of queries for the last 100 users is unchanged.

These results show that WSH is a feasible heuristic. On average, the additional querying has little impact on the initial users. The subsequent improvement for the last users is substantial. However, the problem with WSH is its overall efficiency heavily depends upon the number of users we are processing. This suggests that before we use WSH in practice, it would be very useful to run simulations to see if WSH would be effective for the specific



(a) Uniform distribution

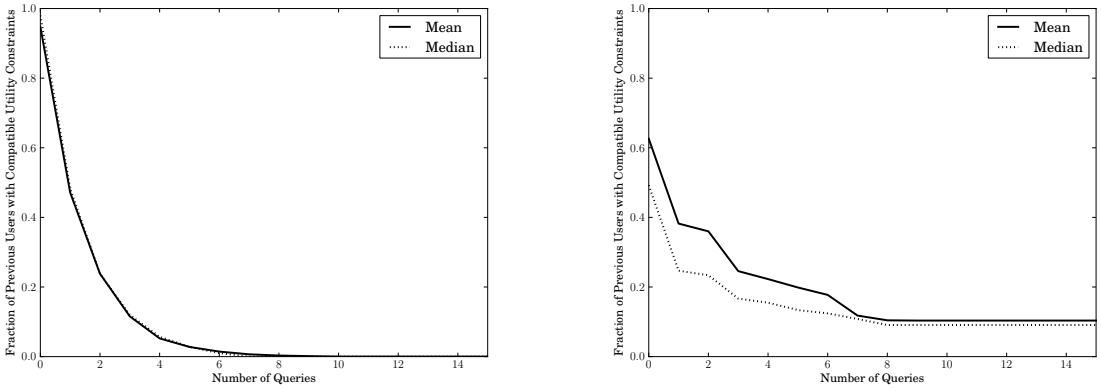
(b) Parametric distribution

Figure 4.9: *Fraction of the previous users with compatible utility constraints after each query. Both the mean and median values are shown. The scenarios were based on 7 outcomes. In Figure 4.9(a), the users' utility values were chosen uniformly at random. This is the worst case in terms of maximizing the number of users with compatible utility constraints. In Figure 4.9(b), the users' utility values were chosen using the parametric function x^r with r chosen uniformly at random between 0 and 1.*

scenario.

While these results show that lPrMMR can improve the performance over MMR, we see that this improvement quickly disappears as we decrease the probability of error. A natural question is how many users do we need to have processed before we see improvements in the performance even with low probabilities of error. While we could create a massive database of users, we found a more efficient way to investigate this problem was to artificially create users throughout the elicitation process. Specifically, after each query, we create a fixed number of users whose utility constraints are compatible with the current user's constraints.

We next investigated how the number of compatible users changes with each additional query. Figure 4.9 shows the mean fraction of the previous users whose utility constraints are compatible with the current user's utility constraints after each query. For example, if for the 11th user, 0.5 of the previous users' utility constraints are still compatible with the 11th user's utility constraints, then 5 previous users are compatible. We examined two situations. We started with a simple situation with 7 outcomes. In the first situation, the users' utility values were chosen uniformly at random, without a monotonicity constraint.



(a) Uniform distribution

(b) Parametric distribution

Figure 4.10: *Fraction of the previous users with compatible utility constraints after each query. Both the mean and median values are shown. The scenarios were based on 20 outcomes.*

Choosing utility values uniformly at at random spreads the values out and as a result it is easier for two utility constraints to be incompatible with each other. Therefore, we can think of these utility values as being the worst-case scenario. These results are shown in Figure 4.9(a). The best case scenario is when utility values are identical. A "close-to" best-case scenario is when the utility values differ by only a small amount. For our "close-to" best case, we choose utility value according to

$$u(x) = x^{\beta},$$

where $0 \leq \beta \leq 1$ is chosen uniformly at random for each user. These results are shown in Figure 4.9(b). The results show that in the worst case, after 8 queries, there are no compatible users. Even after just 4 queries only 10% of the previous users are compatible. In our best case, even after 15 queries, the fraction of users with compatible utility constraints is still around 18%. This shows how critical the distribution of utility values is to the performance of lPrMMR. The fact that the mean and median are roughly equivalent for all queries shows that we are not dealing with a skewed distribution.

Scenarios with only 7 outcomes are obviously very simple. To see how these results scale up to larger scenarios, we repeated these experiments with 20 outcomes and kept all other parameters the same. The results are shown in Figure 4.10. We see that while

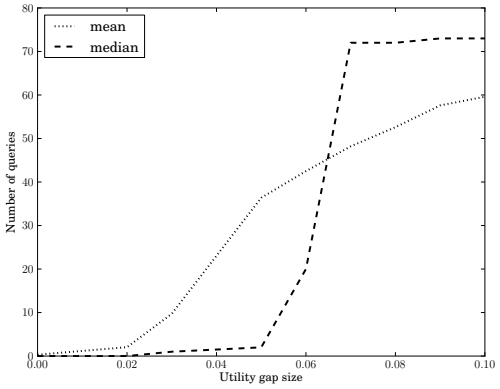


Figure 4.11: *The mean and median number of queries needed to achieve lPrMMR of 0.01 with a probability of error equal to 0.25. For each query, we created 200 compatible users and varied the size of the utility gap for all of their utility constraints. For example, if* $x = 0.03$*, then the size of the utility gap for utility constraints for all users was 0.03.*

the fraction of compatible users decreases at only a slightly faster rate with the uniform distribution, the rate of decrease for a parametric distribution increases significantly. This illustrates that even with a parametric distribution, as we add more outcomes, the number of compatible users can drop noticeably.

Our results are heavily dependent on how accurate the utility constraints we obtain for each user are. Therefore, we next investigated how our results vary depending on the accuracy of the utility constraints. We examined a situation with 20 outcomes and 10 decisions and queried each user using the HLG heuristic, until the lPrMMR was at most 0.01 with $p_{err} = 0.25$ using OPS. Instead of using the utility constraints from previous users, after each query we created 200 simulated users whose utility constraints were compatible with the current user's utility constraints. With this artificial approach, we were able to ensure that the number of compatible users was constant and so any changes were purely the result of the accuracy of the utility constraints. The size of the utility gap was the same for each outcome and varied between 0 and 0.05. The resulting mean and median number of queries needed are shown in Figure 4.11. If the size of the utility gap was 0, *i.e.* we knew the exact utility values, we did not need any queries. As we increased the size of the utility gaps, the mean number of queries quickly increased to 36.4, roughly half of the queries needed with minimax regret. However, the median number of queries only increased to 2. This discrepancy meant that, for most of the users, the loss of accuracy did not make much of a difference. However, for those users where the loss of accuracy made

a difference, the difference was considerable. In fact, for many of the users who needed additional queries, the number of queries needed was very close to the number of queries needed with minimax regret. Thus, we have contrasting results. For most users, lPrMMR is very robust with regards to the accuracy of the utility constraints. However, for the few users where using lPrMMR is not robust, the differences are significant.

After k queries on the utility value of outcome x_i, the size of the resulting utility gap is 2^{-k}. Therefore, it will take 5 queries to reach a utility gap with a size of at most 0.05. For 20 outcomes, this will amount to 90 queries in total (since we do not need to query about the first or last outcome). We can also use Figure 4.11 to find the rough number of users we need to process before we can get a specific level of performance. For example, suppose after a query we had 200 users with compatible utility constraints and each of those users had a utility gap of 0.05. Then the median number of queries needed to process users is 2 queries. Using Figure 4.10 we can estimate that if the user's preferences were uniformly distributed, then after 2 queries only roughly 24% of all users had utility constraints compatible with the current user. This translates into needing 833 users initially. However, if the users' preferences were distributed according to a parametric distribution, then 36% of all users would be compatible with the current user. This translates into needing only 555 users.

4.4.2 Hypothesis Regret Results

Experimental Setup

All of the users' preferences were created using the following probability distribution:

$\mathcal{H}^*$: The values for u_{elect} are given by

$$u_{\text{elect}}(e) = e^{\beta}, \tag{4.50}$$

where $0 \leq \beta \leq 1$ is chosen uniformly at random for each user. We are interested in utility functions of the form in given in Equation 4.50, since it is often used to describe peoples' preferences in experimental settings [59].

To create a challenging experiment, we studied the following set of hypotheses which are feasible with respect to $\mathcal{H}^*$.

$\mathcal{H}_1$: The values for u_{elect} are chosen uniformly at random, without a monotonicity constraint.

$\mathcal{H}_2$**:** The values for u_{elect} are chosen according to Equation 4.50, where $0 \leq r \leq 1$ is chosen according to a Gaussian distribution with mean 0.7 and standard deviation 0.1.

$\mathcal{H}_3$**:** The values for u_{elect} are chosen according to

$$u_{\text{elect}}(x) = x^{\beta} + \Gamma$$

where $0 \leq \beta \leq 1$ is chosen uniformly at random and Γ is chosen uniformly at random between -0.1 and 0.1.

For these experiments we created 200 users whose preferences were created according to $\mathcal{H}^*$. Each user had the same 15 possible cost choices and 15 possible energy outcomes. We asked each user at most 100 queries. Our goal was to achieve a minimax regret of at most 0.01. We rejected hypotheses when $\alpha < 0.01$ which is typically seen as very strong evidence against a hypothesis [69]. For all of our experiments, we chose p in Equation 4.6 to be equal to 1.

Experimental Results

As a benchmark, we first processed users relying just on minimax regret (with and without the monotonicity constraint). The average number of queries needed to solve each user is shown in Table 4.6. We experimented with both the HLG and CS elicitation heuristics (discussed in Section 2.2.3.) Without the monotonicity constraint, the average number of queries was 42.0 using HLG and 66.7 using CS. With the monotonicity constraint, the average was 22.7 using HLG and 53.6 using CS. Table 4.6 also shows the results using hypothesis-based regret with $\mathbb{H} = \{\mathcal{H}^*\}$, *i.e.* what would happen if we knew the correct distribution. In this case, using HLG the average number of queries is 2.4 and using CS the average is 13.3. These results demonstrate that the more we know about the distribution, the better the performance is.

Our next experiments looked at the performance of hypothesis-based regret using the *Reject*(0) heuristic with the following sets for $\mathbb{H}$: $\{\mathcal{H}^*, \mathcal{H}_1\}$, $\{\mathcal{H}^*, \mathcal{H}_2\}$, and $\{\mathcal{H}^*, \mathcal{H}_3\}$. Since, as shown in Table 4.6, the HLG elicitation strategy outperforms the CS strategy for our model, we relied on the HLG strategy. The average number of queries needed to reject a hypothesis, as shown in Table 4.7, was 23.7, 2.4 and 12.9 for $\mathcal{H}_1$, $\mathcal{H}_2$, and $\mathcal{H}_3$, respectively. Both $\mathcal{H}_1$ and $\mathcal{H}_3$ overestimate the actual regret, resulting in an increase in the number of queries needed. While $\mathcal{H}_2$ is not identical to $\mathcal{H}^*$, for our simulations, the regret estimates provided by these two hypotheses are close enough that there is no increase in the number of queries when we include $\mathcal{H}_2$ in $\mathbb{H}$. We were unable to reject any

Regret	HLG	CS
Minimax	42.0	66.7 (135 users not solved)
Minimax with monotonicity	22.7	53.6 (143 users not solved)
$hPrMMR$ with $\mathbb{H} = \{\mathcal{H}^*\}$	2.4	13.3

Table 4.6: *The mean number of queries needed to process a user using either the HLG or CS strategy based on different models of regret. The averages are based on only those users we were able to solve, i.e. obtain a regret of at most 0.01.*

$\mathbb{H}$	Mean
$\{\mathcal{H}^*, \mathcal{H}_1\}$	24.7
$\{\mathcal{H}^*, \mathcal{H}_2\}$	2.4
$\{\mathcal{H}^*, \mathcal{H}_3\}$	12.9

Table 4.7: *Mean number of queries using hPrMMR with the $Reject(0)$ heuristic for different hypotheses sets.*

of the incorrect hypotheses using $Reject(0)$. However, we know from Proposition 3 that our regret calculations are still correct.

We next experimented with the $Reject(1)$ heuristic, again using HLG elicitation heuristics. We tested the same sets of hypotheses for $\mathbb{H}$ as in Table 4.7 and the results are shown in Table 4.8. We were able to reject $\mathcal{H}_1$ after 5 users, which reduced the overall average number of queries to 7.4 when $\mathbb{H} = \{\mathcal{H}^*, \mathcal{H}_1\}$. Thus, we can easily differentiate $\mathcal{H}_1$ from $\mathcal{H}^*$ and doing so improves the overall average number of queries. With the additional querying in $Reject(1)$, we were able to quickly reject $\mathcal{H}_2$. However, since including $\mathcal{H}_2$ did not increase the average number of queries, there is no gain from rejecting $\mathcal{H}_2$ and as a result, the average number of queries rises to 8.29. It took 158 users to reject $\mathcal{H}_3$. As a result, the average number of queries increased to 80.0. This means it is relatively difficult to differentiate $\mathcal{H}_3$ from $\mathcal{H}^*$. In this case, while including $\mathcal{H}_3$ in $\mathbb{H}$ increases the average number of queries, we would be better off not trying to reject $\mathcal{H}_3$ when processing only 200 users.

Finally, we experimented with $\mathbb{H} = \{\mathcal{H}^*, \mathcal{H}_1, \mathcal{H}_2, \mathcal{H}_3\}$ using $Reject(n)$ with different values of n. The results are shown in Table 4.9. With $n = 0$ we are unable to reject any of the incorrect hypotheses, however the average number of queries is still considerably lower

$\mathbb{H}$	Mean	Number of users needed to reject hypothesis
$\{\mathcal{H}^*, \mathcal{H}_1\}$	7.4	5
$\{\mathcal{H}^*, \mathcal{H}_2\}$	8.3	11
$\{\mathcal{H}^*, \mathcal{H}_3\}$	80.0	158

Table 4.8: *Mean number of queries using hPrMMR and Reject*(1) *heuristic for different hypotheses sets.*

$n =$	Mean	Number of users needed to reject $\mathcal{H}_1$,$\mathcal{H}_2$,$\mathcal{H}_3$
0	26.0	NR,NR,NR
1	15.0	5,NR,NR
2	18.5	5,11,NR
3	80.0	5,11,158

Table 4.9: *Mean number of queries for using hPrMMR using the Reject*(n) *heuristic with varying values for* n*. We also show that number of users to reject each hypothesis. The hypotheses set is* $\mathbb{H} = \{\mathcal{H}^*, \mathcal{H}_1, \mathcal{H}_2, \mathcal{H}_3\}$*. NR stands for not rejected.*

than for minimax regret. With $n = 1$ we are able to quickly reject $\mathcal{H}_1$ and, as a result, the average number of queries decreases to 15.0. For $n = 2$ we are able to also reject $\mathcal{H}_2$. However, $\mathcal{H}_2$ takes longer to reject and since $\mathcal{H}_2$ does not increase the number of queries, using *Reject*(2), means the average number of queries rises to 18.5. Finally, with $n = 3$, we are able to reject $\mathcal{H}_3$ as well as $\mathcal{H}_1$ and $\mathcal{H}_2$. While having $\mathcal{H}_3$ in $\mathbb{H}$ increases the number of queries, rejecting $\mathcal{H}_3$ is difficult enough that the average number of queries rises to 80.0.

These experiments show how hypothesis-based regret outperforms minimax regret. While this is most noticeable when we are certain of the correct hypothesis, our approach continues to work well with multiple hypotheses. The *Reject*(n) heuristic is effective at rejecting hypotheses, improving the long term performance of hypothesis-based regret.

4.5 Conclusion

In this chapter, we introduced two new methods for giving an upper bound on the regret from each decision. Both methods provide a probabilistic upper bound on the best worst-case regret. These methods are especially well suited for processing a sequence of users.

The first method, lPrMMR, makes no assumptions about the users' preferences and learns from each of the users. Therefore, as we process more and more users, the performance of lPrMMR increases. The second method, hPrMMR, uses a set of hypotheses about the distribution all the users' preferences are chosen from. As long as one of the hypotheses is correct, the bounds given by hPrMMR are correct. In this case, the correctness of hPrMMR is not affected by incorrect hypotheses. Instead, incorrect hypotheses only result in higher estimates of regret. To improve the performance of hPrMMR, we provided methods for rejecting the incorrect hypotheses.

Experimental results showed that the bounds given by both lPrMMR and hPrMMR are never higher than MMR. The experimental results also showed how the bounds on regret given by both lPrMMR and hPrMMR improve with each additional processed user. The performance of hPrMMR was noticeably better than lPrMMR; however, this is due to the stronger information requirement of hPrMMR.

Chapter 5

Multiattribute Preferences and Preference Elicitation

Multiattribute preferences allow for a natural way to express preferences in complex situations. In the worst case, however, the number of outcomes grows exponentially with respect to the number of attributes. Multiattribute utility independence (MUI) models offer a way to avoid this exponential growth by compactly representing a user's preferences. In this chapter, we discuss how to take advantage of some of these models to improve the preference elicitation process. The main focus of this chapter is to compare the relative compactness of different independence models.

5.1 Preference Elicitation

With or without multiple attributes, there are two key steps for preference elicitation: querying the user and calculating the regret. In Section 2.3.2, we reviewed approaches for querying the user and calculating minimax regret while taking advantage of either Additive Independence or Generalized Additive Independence (GAI) decompositions of preferences. Calculating minimax regret with Additive Indpendence is polynomial in time. However, Additive Independence is strict enough that we can rarely make use of it. While GAI is more useful in practice than Additive Independence, current methods for calculating minimax regret with GAI have, in the worst case, an exponential runtime. In this section, we discuss how Conditional Difference Independence (CDI_r) can achieve a balance between these two MUI models.

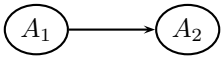


Figure 5.1: A simple example of a CDI_r representation.

We start by discussing how to do preference elicitation based on a CDI_r decomposition. Given a CDI_r decomposition, the goal of eliciting preferences is to bound $u^r_{A_i}(x|Pa(A_i))$ for each attribute. This in turn requires us to bound both λ_f and $v^r_{A_i}(x|f)$ for every attribute A_i and every local outcome $f \in Pa(A_i)$. This can be done simply by using the global and local standard gamble queries discussed in Section 2.3.2.

We next show that, given a CDI_r decomposition, calculating minimax regret is polynomial with respect to the size of the CDI_r representation. Let $\Pr_{d,A}$ be the marginal probability distribution over the attribute A given decision d. For example, if $\mathbb{A} = \{A_1, A_2\}$, then for each outcome $a_1 \in A_1$, we would have the marginal distribution

$$\Pr_{d,A_1}(a_1) = \sum_{a_2 \in A_2} \Pr_d((a_1, a_2)).$$

We also define $C_{A|Pa(A)}$ to be the utility constraints over the local outcomes in A given $Pa(A)$. For example, if $\mathbb{A} = \{A_1, A_2, A_3\}$, $Pa(A_3) = \{A_1, A_2\}$ and $A_1 = \{a_1, a_1'\}$, $A_2 = \{a_2, a_2'\}$ and $A_3 = \{a_3, a_3'\}$ then

$$C_{A_3|Pa(A_3)} = \{C(a_3|\{a_1, a_2\}), C(a_3'|\{a_1, a_2\}), \ldots, C(a_3'|\{a_1', a_2\}), \ldots\}$$

where $C(a_3|\{a_1, a_2\})$ is the utility constraint on the local outcome a_3 given the conditional outcome $\{a_1, a_2\}$.

With these definitions, we calculate pairwise maximum regret, Equation 2.13, as;

$$
\begin{aligned}
PMR(d, d', C) &= \max_{u \in C} \sum_{x \in \mathbb{X}} (\Pr_{d'}(x) - \Pr_{d}(x)) u(x) \\
&= \max_{u \in C} \sum_{x \in \mathbb{X}} (\Pr_{d'}(x) - \Pr_{d}(x)) \sum_{A \in \mathbb{A}} u^r_A(x|Pa(A)) \\
&= \max_{u \in C} \sum_{A \in \mathbb{A}} \sum_{x \in \mathbb{X}} (\Pr_{d'}(x) - \Pr_{d}(x)) u^r_A(x|Pa(A)) \\
&= \max_{u \in C} \sum_{A \in \mathbb{A}} \sum_{a \in A} \left[\Pr_{d'}(a) - \Pr_{d}(a) \right] u^r_A(a|Pa(A)) \qquad (5.1) \\
&= \sum_{A \in \mathbb{A}} \sum_{p \in Pa(A)} \max_{u \in C_{A|p}} \sum_{a \in A} \left[\Pr_{d'}(a \wedge p) - \Pr_{d}(a \wedge p) \right] u(a), \qquad (5.2)
\end{aligned}
$$

where $\Pr_{d'}(a \wedge p)$ is the marginal probability over $A \times Pa(A)$. We can simplfy Equation 5.1 to 5.2 because of a unique property of calculating PMR using a CDI_r representation; the utility values we choose for one node do not affect which utility values we can choose for other nodes. For example, consider the CDI_r representation in Figure 5.1. We calculate the minimax regret by choosing values for $u^r_{A_1}(a)$ and $u^r_{A_2}(a|Pa(A_1))$. However, $u^r_{A_2}(a|Pa(A_1))$ is not dependent on $u^r_{A_1}(a)$, $u^r_{A_2}(a|Pa(A_1))$ is only dependent on the realization of the local oucome for the attribute A_1. Since calculating minimax regret does not require choosing realizations, we can maximize $u^r_{A_1}$ and $u^r_{A_2}(a|Pa(A_1))$ independently of each other.

We can constrast this unique property by considering the related problem of choosing the outcome which maximizes utility using a CDI_r representation. (This problem is analogous to maximizing probabilities in a Bayesian network [11].) For utility maximization, we need to choose the local outcome for each attribute. Since the possible utility values for a child node are dependent on the chosen local outcomes of the parent nodes, we cannot consider each attribute independently.

We illustrate the uniqueness of calculating minimax regret using CDI_r representations with the following definition and example.

Definition 14 (Local Pairwise Maximum Regret)**.** *Given a CDI_r representation and corresponding utility constraints, the* local pairwise maximum regret *(lPMR) between decisions d and d' with respect to the attribute A is given by*

$$
\begin{aligned}
&\max_{u \in C_{A|Pa(A)}} \sum_{p \in Pa(A)} \sum_{a \in A} \left[\Pr_{d'}(a \wedge p) - \Pr_{d}(a \wedge p) \right] u(a), \qquad (5.3) \\
&= \sum_{p \in Pa(A)} \max_{u \in C_{A|p}} \sum_{a \in A} \left[\Pr_{d'}(a \wedge p) - \Pr_{d}(a \wedge p) \right] u(a).
\end{aligned}
$$

A_1	a_1^1	a_2^1	a_3^1
$u^r_{A_1}(A)$	[0,0.3]	[0.6,0.7]	[0.0,0.5]

(a) C_{A_1}

A_2	a_1^2	a_2^2	a_3^2
$u^r_{A_2}(A\|A_1 = a_1^1)$	[0.1,0.8]	[0.3,0.7]	[0.4,1]
$u^r_{A_2}(A\|A_1 = a_2^1)$	[0.4,0.5]	[0.2,0.6]	[0.1,0.2]
$u^r_{A_2}(A\|A_1 = a_3^1)$	[0.3,0.9]	[0.1,0.3]	[0.0,0.5]

(b) $C_{A_2|A_1}$

Table 5.1: Example utility constraints for $u^r_{A_1}(A)$ and $u^r_{A_2}(A|A_1)$ given the CDI$_r$ representation given in Figure 5.1.

A_1	a_1^1	a_2^1	a_3^1
$Pr_d(A_1)$	1/2	1/4	1/4

(a) A_1

A_2	a_1^2	a_2^2	a_3^2
$Pr_d(A_2)$	1/8	5/8	3/8

(b) A_2

Table 5.2: Probability distributions over A_1 and A_2 resulting from the decision d.

Example: Suppose we have the CDI$_r$ representation given in Figure 5.1. This representation has two attributes, A_1 and A_2. The conditional utility values for A_2 are dependent on the realization of A_1. For example, suppose both A_1 and A_2 have three local outcomes, $\{a_1^1, a_2^1, a_3^1\}$ and $\{a_1^2, a_2^2, a_3^2\}$, respectively. We have two decisions, d_1 and d_2 as described in Tables 5.2 and 5.3, respectively. A set of example utility constraints are shown in Table 5.1.

To calculate the PMR between d and d' with respect to the utility constraints in Table 5.1, we need to calculate the lPMR with respect to both A_1 and A_2. In this example, we choose to start by calculating the lPMR with respect to A_1. We note that we could just as easily start with A_2 without having to restructure our CDI$_r$ representation. We can

A_1	a_1^1	a_2^1	a_3^1
$Pr_{d'}(A_1)$	1/6	1/3	1/2

(a) A_1

A_2	a_1^2	a_2^2	a_3^2
$Pr_{d'}(A_2)$	1/7	3/7	3/7

(b) A_2

Table 5.3: Probability distributions over A_1 and A_2 resulting from the decision d'.

rewrite Equation 5.3 as

$$\begin{aligned} lPMR_{A_1}(d,d') &= \sum_{a\in A_1}\left[\Pr_{d',A_1}(a) - \Pr_{d,A_1}(a)\right]\begin{cases} u^{\max}_{A_1}(a) & \text{if } \Pr_{d',A_1}(a) > \Pr_{d,A}(a_1) \\ u^{\min}_{A_1}(a) & \text{otherwise,}\end{cases} \\ &= (1/6-1/2)u^{\min}_{A_1}(a_1^1) + (1/3-1/4)u^{\max}_{A_1}(a_2^1) + (1/2-1/4)u^{\max}_{A_1}(a_3^1) \\ &= -\frac{1}{3}0 + \frac{1}{12}0.7 + \frac{1}{4}0.5 \\ &= 0.183. \end{aligned}$$

To calculate the lPMR with respect to A_2, we need to calculate $lPMR_{A_2|a_1^1}$, $lPMR_{A_2|a_2^1}$ and $lPMR_{A_2|a_3^1}$.

$$\begin{aligned} &lPMR_{A_2|a_1^1}(d,d') \\ &\quad= \left(\frac{1}{6}\cdot\frac{1}{7} - \frac{1}{2}\cdot\frac{1}{8}\right)u^{\min}_{A_2}(a_1^2|a_1^1) + \left(\frac{1}{6}\cdot\frac{3}{7} - \frac{1}{2}\cdot\frac{5}{8}\right)u^{\min}_{A_2}(a_2^2|a_1^1) + \left(\frac{1}{6}\cdot\frac{3}{7} - \frac{1}{2}\cdot\frac{1}{8}\right)u^{\max}_{A_2}(a_1^2|a_1^1) \\ &\quad= -\frac{13}{336}\cdot 0.1 - \frac{27}{112}\cdot 0.3 + \frac{1}{112}\cdot 1 \\ &\quad= 0.0673 \end{aligned}$$

$$\begin{aligned} &lPMR_{A_2|a_2^1}(d,d') \\ &\quad= \left(\frac{1}{3}\cdot\frac{1}{7} - \frac{1}{4}\cdot\frac{1}{8}\right)u^{\max}_{A_2}(a_1^2|a_2^1) + \left(\frac{1}{3}\cdot\frac{3}{7} - \frac{1}{4}\cdot\frac{5}{8}\right)u^{\min}_{A_2}(a_2^2|a_2^1) + \left(\frac{1}{3}\cdot\frac{3}{7} - \frac{1}{4}\cdot\frac{1}{8}\right)u^{\max}_{A_2}(a_1^2|a_2^1) \\ &\quad= \frac{11}{672}\cdot 0.5 - \frac{3}{224}\cdot 0.2 + \frac{25}{224}\cdot 0.2 \\ &\quad= 0.0278 \end{aligned}$$

outcome	u^{min}	$u^{\max}$	$\Pr_d$	$\Pr_{d'}$
a_1^1, a_1^2	0.1	1.1	1/16	1/42
a_1^1, a_2^2	0.3	1.0	5/16	3/42
a_1^1, a_3^2	0.4	1.3	3/16	3/42
a_2^1, a_1^2	1.0	1.2	1/32	1/21
a_2^1, a_2^2	0.8	1.3	5/32	3/21
a_2^1, a_3^2	0.7	0.9	3/32	3/21
a_3^1, a_1^2	0.3	1.4	1/32	1/14
a_3^1, a_2^2	0.1	0.8	5/32	3/14
a_3^1, a_3^2	0.0	1.0	3/32	3/14

Table 5.4: Utility constraints from Table 5.1 converted to utility constraints over global outcomes.

$$
\begin{aligned}
&lPMR_{A_2|a_3^1}(d, d') \\
&\quad = \left(\frac{1}{2}\cdot\frac{1}{7} - \frac{1}{4}\cdot\frac{1}{8}\right) u_{A_2}^{\max}(a_1^2|a_3^1) + \left(\frac{1}{2}\cdot\frac{3}{7} - \frac{1}{4}\cdot\frac{5}{8}\right) u_{A_2}^{\max}(a_2^2|a_3^1) + \left(\frac{1}{2}\cdot\frac{3}{7} - \frac{1}{4}\cdot\frac{1}{8}\right) u_{A_2}^{\max}(a_1^2|a_3^1) \\
&\quad = \frac{9}{224}\cdot 0.9 + \frac{13}{224}\cdot 0.3 + \frac{41}{224}\cdot 0.5 \\
&\quad = 0.1451
\end{aligned}
$$

Therefore,

$$
\begin{aligned}
PMR &= lPMR_{A_1}(d, d') + lPMR_{A_2|a_1^1}(d, d') + lPMR_{A_2|a_2^1}(d, d') + lPMR_{A_2|a_3^1}(d, d') && (5.4) \\
&= 0.4232 && (5.5)
\end{aligned}
$$

We next compare the above derivation against the standard PMR calculation from Equation 2.13. To do so, we first convert the utility constraints in Table 5.1 into utility constraints over the global outcomes. The resulting utility constraints are shown in Table 5.4. There are two things to note about the constraints in Table 5.4. First, several of the upper bound constraints are above 1. For example, $u^{\max}(a_3^1, a_1^2) = 1.4$. What this means is that at least one of the constraints $u_{A_1}^{\max}(a_3^1)$ and $u_{A_2}^{\max}(a_1^2|a_3^1)$ is too high. However, we have no way of knowing which is the constraint that is too high. We also note that (a_3^1, a_3^2) is the only constraint for which the minimum utility value is 0. This means that $x_\perp = (a_3^1, a_3^2)$. As a result, we can set both $u_{A_1}^{\max}(a_3^1)$ and $u_{A_2}^{\max}(a_3^2)$ to be zero. Both of these cases illustrate how converting from a CDI_r representation into a non-multiattribute setting can provide additional information.

The derviation for $PMR(d, d')$ according to Equation 2.13 is as follows. To provide a direct comparison against the result from Equation 5.5, we do not alter the constraints from Table Table 5.1.

$$
\begin{aligned}
PMR(d, d') &= \left(\frac{1}{42} - \frac{1}{16}\right) u^{\min}((a_1^1, a_1^2)) + \left(\frac{3}{42} - \frac{5}{16}\right) u^{\min}((a_1^1, a_2^2)) + \left(\frac{3}{42} - \frac{3}{16}\right) u^{\min}((a_1^1, a_3^2)) \\
&+ \left(\frac{1}{21} - \frac{1}{32}\right) u^{\max}((a_2^2, a_1^2)) + \left(\frac{3}{21} - \frac{5}{32}\right) u^{\min}(a_2^2, a_2^2)) + \left(\frac{3}{21} - \frac{3}{32}\right) u^{\max}((a_2^1, a_3^2)) \\
&+ \left(\frac{1}{14} - \frac{1}{32}\right) u^{\max}((a_3^1, a_1^2)) + \left(\frac{3}{14} - \frac{5}{32}\right) u^{\max}((a_3^1, a_2^2)) + \left(\frac{3}{14} - \frac{3}{32}\right) u^{\max}((a_3^1, a_3^2)) \\
&= -\frac{13}{336} u^{\min}((a_1^1, a_1^2)) - \frac{27}{112} u^{\min}((a_1^1, a_2^2)) - \frac{13}{112} u^{\min}((a_1^1, a_3^2)) \\
&+ \frac{11}{672} u^{\max}((a_2^2, a_1^2)) - \frac{3}{224} u^{\min}(a_2^2, a_2^2)) + \frac{11}{224} u^{\max}((a_2^1, a_3^2)) \\
&+ \frac{9}{224} u^{\max}((a_3^1, a_1^2)) + \frac{13}{224} u^{\max}((a_3^1, a_2^2)) + \frac{27}{224} u^{\max}((a_3^1, a_3^2)) \\
&= -\frac{13}{336} \cdot 0.1 - \frac{27}{112} \cdot 0.3 - \frac{13}{112} \cdot 0.4 \\
&+ \frac{11}{672} \cdot 1.2 - \frac{3}{224} \cdot 0.8 + \frac{11}{224} \cdot 0.9 \\
&+ \frac{9}{224} \cdot 1.4 + \frac{13}{224} \cdot 0.8 + \frac{27}{224} 1.0 \\
&= -0.1226 + 0.0531 + 0.6920 \\
&= 0.6225
\end{aligned}
$$

Using Equation 2.13 instead of Equation 5.5 to calculate PMR resulted in a significantly higher than; 0.6225 verses 0.4232. □

The lPMR can be calculated in linear time with respect to the size of $A \times Pa(A)$. Let $Pa_{\max}$ be the conditional set with the largest cardinality, that is

$$Pa_{\max} = \arg \max_{Pa(A)|A \in \mathbb{A}} |Pa(A)|.$$

Therefore, the overall PMR can be calculated in $O(|\mathbb{A}| \cdot |A \times Pa_{\max}|) = O(n \cdot |A \times Pa_{\max}|)$. In the best case where additive independence holds, $Pa_{\max} = \emptyset$, then the runtime becomes $O(n|A|) = O(nm)$. In the worst case where no independence holds, then assuming the attributes are ordered $\{A_1, \ldots, A_n\}$, $Pa_{\max} = Pa(A_n) = \{A_1 \times \ldots \times A_{n-1}\}$ which gives a runtime of $O(m^n)$. This complexity means that we can calculate minimax regret using a

CDI_r decomposition in polynomial time with respect to the size of the CDI_r decomposition. Thus, we see that using a CDI_r decomposition allows us to calculate minimax regret as efficently as if we were using a GAI decomposition [10, 13].

5.2 Comparing Decompositions

In this section, we compare how compactly preferences can be represented using different types of utility independence. Our focus is on comparing CDI_r to other MUI models; specifically Conditionally Additive Independence (CAI) and GAI, since there are well-established methods for calculating minimax regret using both of these models. We also compare Conditional Utility Independence (CUI) against both GAI and CDI_r.

Converting CDI_r to GAI

To convert from CDI_r to GAI, we take every marginal utility function $u^r_{A_i}(A_i|Pa(A_i))$ and convert it into a LVF $v_{A_i \times Pa(A_i)}(x)$. Since both the marginal utility function and the LVF will have a complexity of $m^{1+|Pa(A_i)|}$, GAI decompositions are always as compact as CDI_r decompositions.

Converting GAI to CDI_r

We next show that, in the worst case, GAI decompositions may be exponentially more compact than the corresponding CDI_r decompositions.

To convert GAI into CDI_r, we rely on the graphical form of CDI_r, given in Section 2.3.1. The end result of our conversion will be a directed graph $G = (\mathbb{A}, \mathbb{E})$. The complexity of the resulting CDI_r decomposition will be determined by the maximum indegree in G. Specifically, if the maximum indegree in G is j, then the decomposition will have a complexity of $O(m^{j+1})$. This conversion is done by converting each LVF in the GAI decomposition into a subgraph of G using the following two definitions and lemma.

Definition 15 (Skeleton CDI_r graph)**.** *A skeleton CDI_r graph is an undirected graph based on a CDI_r graph with all the directions on the edges removed. Given the CDI_r decomposition $\{A_0, Pa(A_1), \ldots, Pa(A_n)\}$, we create the skeleton graph $G^{skeleton}_{CDI}(\mathbb{A}, \mathbb{E})$ by including the edge (A_i, A_j) if and only if either $A_i \in Pa(A_j)$ or $A_j \in Pa(A_i)$.*

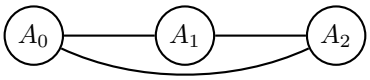


Figure 5.2: *An example of a skeleton CDI$_r$ graph for the LVF $v_{\{A_0,A_1,A_2\}}(x)$. Since the skeleton is a clique any ordering of the attributes will result in the same complexity for the CDI$_r$ representation. For example, if we choose the ordering $\{A_0, A_1, A_2\}$, A_2 will have an indegree of 2, A_1 will have an indegree of 1 and A_0 will have indegree of 0. If we choose the alternative ordering $\{A_0, A_2, A_1\}$, A_1 will have an indegree of 2, A_2 will have an indegree of 1 and A_0 will have an indegree of 0. Either way, the resulting CDI$_r$ representation will have a complexity of $\Theta(m^3)$ where m is the number of local outcomes.*

Definition 16 (Clique)**.** *A clique is a fully connect subgraph of an undirected graph.*

Lemma 8. *Suppose that the LVF $v_F(x)$ where $F^{att} = \{A_0, \ldots, A_l\}$ cannot be factored. Then the skeleton CDI$_r$ subgraph corresponding to $v_F(x)$ must be a clique.*

Proof. For simplicity, we ignore other LVFs in the GAI decompositions. We start off by showing that given a clique skeleton, we are able to choose an ordering of the attributes such that the resulting CDI$_r$ subgraph is able to represent $v_F(x)$. We then show that we are unable to remove any edges in the skeleton subgraph and still be able to represent $v_F(x)$.

If the optimal skeleton subgraph is a clique, all possible orderings of the attributes are isomorphic to each other, *i.e.* the ordering does not affect the complexity of the representation. For example, consider a LVF over the attributes $\{A_0, A_1, A_2\}$. The clique skeleton over these three attributes is shown in Figure 5.2. If we convert the skeleton into a CDI$_r$ graph by imposing the ordering $\{A_0, A_1, A_2, \}$, A_2 will have an indegree of 2, A_1 will have an indegree of 1 and A_0 will have an indegree of 0. If we choose a different ordering, say $\{A_2, A_1, A_0\}$, A_0 will have an indegree of 2, A_1 will have an indegree of 1, A_1 will have an indegree of 0. In fact, imposing any ordering on the nodes in Figure 5.2 will result in one node having an indegree 2, one node having an indegree of 1 and one node having an indegree of 0. As a result, the ordering of the attributes has no effect on the complexity of the resulting CDI$_r$ graph.

Therefore, without loss of generality, we assume the ordering $\{A_0, \ldots, A_n\}$. By definition of a skeleton CDI$_r$ graph, the only way to create a clique skeleton is by setting $Pa(A_i) = \{A_0, \ldots, A_{i-1}\}$. For notational simplicity, we let $u^r_{0,1,2}(x) = u^r_{A_0 \times A_1 \times A_2}(x)$, *etc.*

As a result, the sum of the conditional utilities for all attributes in F^{att} is

$$\begin{aligned}
&\sum_{i=0}^{l} v_i^r(x|\{A_0,\ldots,A_{i-1}\}) \\
&= v_0^r(x) + (v_{0,1}^r(x) - v_0^r(x)) + \ldots + (v_{0,\ldots,n-1,n}^r(x) - v_{0,\ldots,n-1}^r(x)) && (5.6)\\
&= v_{0,\ldots,l}^r(x) && (5.7)\\
&= v_F(x).
\end{aligned}$$

Therefore, with the clique skeleton and the ordering $\{A_0,\ldots,A_l\}$, we are able to represent $v_F(x)$. The above derivation works because Equation 5.6 is a telescoping series where all but one term cancel out.

Suppose we wished to find a simpler skeleton subgraph that could still represent $v_F(x)$. For example, we might want to remove the edge between A_{n-1} and A_n. We would do this by setting $Pa(A_n) = \{A_0,\ldots,A_{n-2}\}$. As a result, we would get the following

$$\begin{aligned}
&\sum_{i=0}^{n-1} v_i^r(x|\{A_0,\ldots,A_{i-1}\}) + v_n^r(x|\{A_0,\ldots,A_{n-2}\}) \\
&= v_{0,\ldots,n-1}^r(x) + v_{0,\ldots,n-2,n}^r(x) - v_{0,\ldots,n-2}^r(x) && (5.8)
\end{aligned}$$

Since Equations 5.7 and 5.8 are not the same, Equation 5.8 is not equal to $v_F(x)$. Therefore, the skeleton corresponding to $v_F(x)$ must be a clique. □

The complexity of storing the conditional utility values for one attribute is equal to the attribute's indegree plus one, *e.g.* if an attribute has an indegree of 3, the complexity of storing the conditional utility values for that attribute is m^4 where m is the number of local outcomes.

Definition 17 (Sink)**.** *A sink is a node in a directed graph with no outgoing edges.*

Any topological ordering of the nodes will result in at least one sink, specifically the last node according to the ordering. For the feature F, since the resulting skeleton is a clique, all nodes are connected which means there must be exactly one sink. The indegree for the sink node is $m^{|F^{att}|-1}$ which means the complexity for storing the conditional utility values at the sink is $m^{|F^{att}|}$ (ignoring possible overlapping features). This means that the complexity from using either a GAI or CDI$_r$ decomposition to store a single feature is the same.

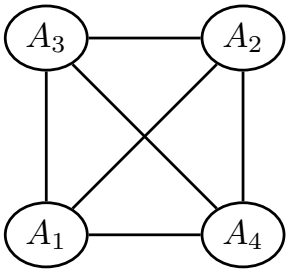


Figure 5.3: *A possible skeleton CDI$_r$ graph over the four attributes $\{A_1, A_2, A_3, A_4\}$. All attribute orderings lead to a complexity of $O(m^4)$. However, multiple GAI decompositions of varying complexity map to this CDI$_r$ decomposition.*

Any increase in complexity must be the result of converting overlapping LVFs. We illustrate this increase in complexity with an example. Figure 5.3 shows an undirected CDI$_r$ graph for some utility function u over 4 attributes. In this case, the graph is a *clique* or a *complete graph*. As a result, all orderings are isomorphic to each other. This means that all orderings induce a directed graph that is able to represent u. In all of these orderings, there will always be a node with indegree of 3 which gives a CDI$_r$ complexity of $O(m^4)$. The problem is that there are multiple GAI decompositions which all lead to this graph; *e.g.*

$$u(x) = u_{A_1 \times A_2}(x) + u_{A_1 \times A_3}(x) + u_{A_1 \times A_4}(x) + u_{A_2 \times A_3}(x) + u_{A_2 \times A_4}(x) + u_{A_3 \times A_4}(x)$$

and

$$u(x) = u_{A_1 \times A_2 \times A_3}(x) + u_{A_1 \times A_2 \times A_4}(x) + u_{A_1 \times A_3 \times A_4}(x) + u_{A_2 \times A_3 \times A_4}(x)$$

both lead to the CDI$_r$ decomposition in Figure 5.3. The complexity of the first decomposition is $6m^2$ and the complexity of the second is $4m^3$. The problem is that the graph in Figure 5.3 is unable to distinguish between these two decompositions. In fact, according to Figure 5.3, u cannot be decomposed at all using a CDI$_r$ representation.

We generalize this problem by considering a fully connected CDI$_r$ graph with n attributes. Since all orderings are isomorphic to each other, without loss of generality, we assume that the ordering is $\{A_1, \ldots, A_n\}$. This means that the indegree for A_1 is 0, for A_2 is 1, *etc.* The complexity of A_1 is m, the complexity of A_2 is m^2, *etc.*. This gives an

Algorithm 3 A greedy algorithm for converting GAI representations into CDI_r representations.

- Convert the GAI decomposition into an undirected CDI_r graph.
- Rank attributes by their degree in the undirected CDI_r graph in descending order.
- Order attributes according to their rank, *e.g.* the attribute with the highest degree is first. Ties are broken at random.

overall complexity of

$$\begin{aligned}
&\sum_{1 \leq i \leq n} m^i \\
&= \frac{m^{n+1} - 1}{m - 1} - 1 \\
&\approx m^n. \qquad (5.9)
\end{aligned}$$

There are many different GAI decompositions which map to this one CDI_r decomposition. Of all these decompositions, the one with the lowest complexity is

$$u(X) = \sum_{\substack{\{A_i, A_j\} \subseteq A \\ i<j}} u_{A_i, A_j}(X).$$

The complexity of this decomposition is

$$\frac{n(n-1)}{2} m^2,$$

which is exponentially more compact than Equation 5.9.

We are also interested in giving an upper bound on the complexity of a CDI_r decomposition in terms of the complexity of the original GAI decomposition. To do so, we used Algorithm 3. As we did with finding the maximum clique, we start by converting the GAI decomposition into a skeleton CDI_r graph using a clique representation for each LVF. Attributes are then inversely ordered by their degree, *e.g.* the attribute with the highest degree is ranked first, *etc.* Ties are broken at random. The combination of the undirected CDI_r graph and the ordering of the attributes gives a CDI_r representation. The complexity of any ordering given by this algorithm is an upper bound on the complexity of the optimal CDI_r decomposition. Since this is a randomized algorithm, we can repeat it multiple times to try and find a better ordering.

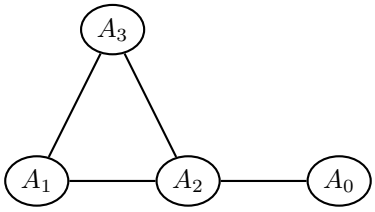


Figure 5.4: *The skeleton CDI$_r$ graph representation of Equation 5.10.*

Example: Suppose we have 4 attributes, $\{A_0, A_1, A_2, A_3\}$, and the following GAI representation of u,

$$u(x) = \lambda_{A_0,A_1} v_{A_0 \times A_2}(x) + \lambda_{A_1,A_2,A_3} v_{A_1 \times A_2 \times A_3}(x). \tag{5.10}$$

We use a clique representation for each LVF in Equation 5.10. The resulting skeleton CDI$_r$ decomposition is shown in Figure 5.4. The degrees of the attributes in Figure 5.4 are $\{1, 2, 3, 2\}$, respectively. Then one possible order which could result from our algorithm is $\{A_2, A_3, A_1, A_0\}$. Since there are two attributes with degree 2, we could also have the order $\{A_2, A_1, A_3, A_0\}$. In this case, both orderings result in CDI$_r$ representations with the same complexity.

Converting CAI to CDI$_r$

We next show that CDI$_r$ is (asymptotically with respect to both m and n) as compact as CAI. To do so, we provide an algorithm, shown in Algorithm 4, to convert from a CAI decomposition to a CDI$_r$ decomposition. We first convert the CAI decomposition into a GAI network. By Corollary 1, there will exist no cycles in the GAI network which are not part of a clique. We also create a skeleton CDI$_r$ graph over the attributes $\mathbb{A}$ where each LVF in the CAI decompositions is represented by a clique.

Finally, we need to choose an ordering over the attributes. We choose any node in the GAI network and use either a *breadth first search* (BFS) or *depth first search* (DFS) to create an ordering of the features, $\{F_1, \ldots, F_l\}$. We iterate through the features, assigning an order to each attribute the first time we encounter it. Some of the attributes will already have been ordered when we converted a previous feature. The order chosen for the remaining (*i.e.* new) attributes in the feature does not matter, as long as they are ordered

Algorithm 4 An algorithm for converting a CAI decomposition of a utility function into a CDI_r decomposition which does not increase the complexity of the decomposition.

Require: A CAI decomposition of a utility function u.

Ensure: A CDI_r decomposition of u based on an directed graph G. The directed graph is based on an undirected graph $G' = (\mathbb{A}, \mathbb{E})$ and a topological ordering of the attributes.

- Convert the CAI decomposition into a GAI graph.
- Using either a BFS or DFS rooted at any feature in the GAI graph, create an ordering of the features, $\mathbb{F} = \{F_1, \ldots, F_l\}$.

for $F_i \in \mathbb{F}$ **do**

- For all attributes in F_i that have not already been assigned an order, choose any ordering which preserves the rank of the attributes which have already been ordered, that is, rank the new attributes after the old ones.

end for

after the already encountered attributes. (If we have an unconnected GAI network, we just repeat the search over each component.)

Lemma 9. *The ordering of the attributes given by Algorithm 4 creates a CDI_r decomposition with asymptotically (as $n \to \infty$) the same complexity as the original CAI decomposition.*

Proof. We show that for each attribute, the indegree for that attribute is at most equal to the size of the feature where we first encountered that attribute, minus 1. For example, if we encountered attribute A_i while processing the feature F_k, then the indegree for A_i in our CDI_r graph is at most $|F_k| - 1$.

For this proof, it is helpful to think that an edge does not exist until we have processed the corresponding feature, *e.g.* if there is an edge between A_i and A_j due to the feature F_k, then we pretend that edge does not exist until we process F_k, even if both A_i and A_j have both been assigned ranks before. Let F_0 be the root feature chosen during our BFS or DFS.

When we first encounter A_i and assign it a rank, the indegree of A_i is at most $|F_k| - 1$. The indegree of A_i can only increase if, while processing another feature, say F_s, we encounter another attribute, say A_j, which has a lower rank than A_i and does not already have a directed edge into A_i. We use proof by contradiction and assume such an attribute A_j exists. For A_j to have been given a rank without there being an edge from A_j to A_i, there must exist a sequence of features $F_0, F_{s_1}, \ldots, F_s$ which does not include F_k and in which every feature includes A_j and none include A_i. Similarly, there must exist the sequence

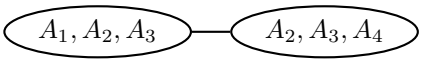


Figure 5.5: *The GAI graph representation of the CAI decomposition given in Equation 5.11.*

$F_0, F_{k_1}, \ldots, F_k$ where every feature includes A_i but none include A_j. This means that we can construct the cycle $F_0, F_s, \ldots, F_s, F_k, F_{k-1}, \ldots, F_{k_1}, F_0$ where the overlap between features changes, which contradicts Corollary 1. Therefore, A_j cannot exist.

Therefore, the complexity of representing the conditional utility at each attribute is $O(m^{|F_k|})$. If we let $F_{\max} = \arg\max_k |F_k|$, then the overall CDI_r decomposition has a complexity of $O(m^{|F_{\max}|})$ which is also the complexity of the initial CAI decomposition. □

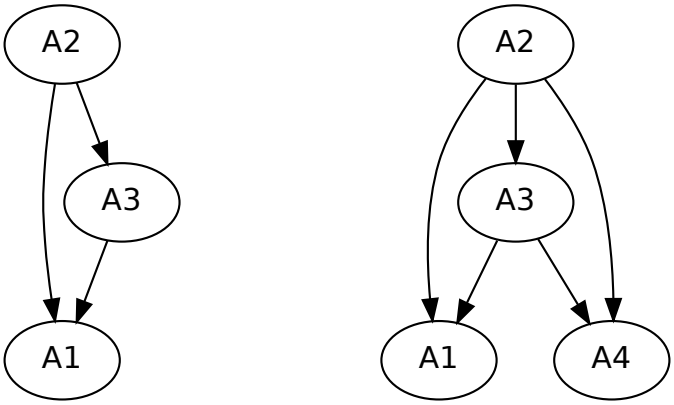
Example: Suppose we have the following CAI decomposition of a utility function which we want to turn into a CDI_r decomposition:

$$u(x) = u_{A_1 \times A_2 \times A_3}(x) + u_{A_2 \times A_3 \times A_4}(x). \tag{5.11}$$

We start by first converting Equation 5.11 to the GAI graph decomposition shown in Figure 5.5. Our algorithm works with any ordering of the features in Figure 5.5 based on a BFS or DFS. In our example, any ordering is the result of a BFS. Therefore, suppose $F_1 = \{A_1, A_2, A_3\}$ and $F_2 = \{A_2, A_3, A_4\}$. We next need to decide on an ordering of the attributes in F_1. According to our algorithm, any ordering will work. Therefore, we choose the ordering A_1, A_3, A_2. Using this ordering, we create the CDI_r graph representation of F_1 as shown in Figure 5.6(a) We next convert F_2. Since the ordering of A_2 and A_3 has already been decided, the ordering of A_4 is automatically determined. The resulting CDI_r decomposition is shown in Figure 5.6(b). Note that the complexity of the CAI decomposition is $2m^3$ and the complexity of the CDI_r decomposition is $2m^3 + m^2 + m$.

Given the ordering of the features in the GAI graph in Figure 5.5, the attribute A_4 will always be ordered last. However, according to Algorithm 4, the ordering of the attributes A_1,A_2 and A_3 does not matter. To demonstrate this flexibility, we consider another example of converting Equation 5.11 into a CDI_r representation where we choose the ordering A_2, A_3, A_1, A_4. The resulting conversion is shown in Figure 5.7. □

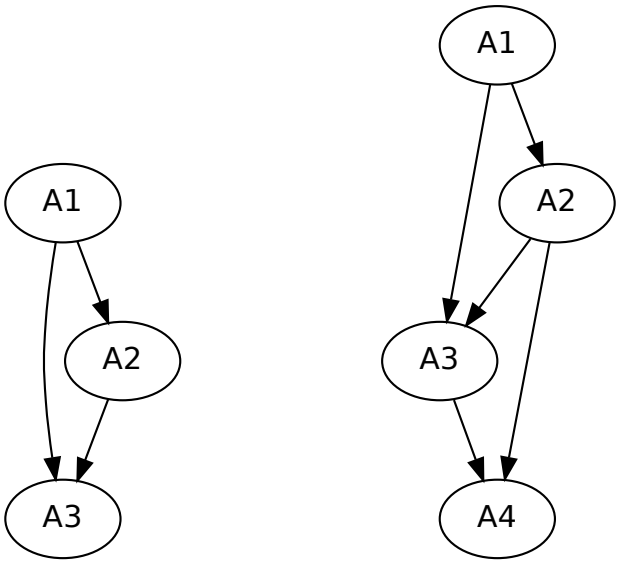
There is a definite similarity between converting a CAI (or GAI) representation into a CDI_r graph and creating a Bayesian network. Algorithms exist for constructing Bayesian networks which first create a skeleton of the Bayesian network and orient the directions of the edges based on observed data [51]. This similarity is not too surprising given the



(a) The CDI_r decomposition of F_1.

(b) The CDI_r representation of $F_1 + F_2$.

Figure 5.6: *The CDI_r graph decomposition of the CAI decomposition given in Equation 5.11 based on the ordering $F_1 = \{A_1, A_2, A_3\}$ and $F_2 = \{A_2, A_3, A_4\}$. Figure 5.6(a) shows the CDI_r decomposition of just F_1. Figure 5.6(b) shows the final decomposition.*



(a) The CDI_r representation of F_1.

(b) The CDI_r representation of $F_1 + F_2$.

Figure 5.7: *A second example of a CDI_r representation of the CAI factorization given in Equation 5.11 based on the ordering $F_1 = \{A_2, A_3, A_1\}$ and $F_2 = \{A_2, A_3, A_4\}$. Figure 5.7(a) shows the CDI_r representation of just F_1. Figure 5.7(b) shows the final representation.*

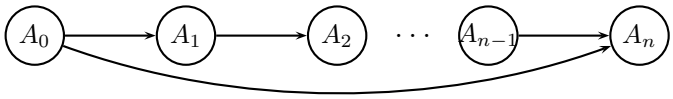


Figure 5.8: *A CDI$_r$ graphical representation with a complexity of* $O(m^3 + (n-1)m^2 + m)$.

strong connections between CDI$_r$ graphs and Bayesian networks [11]. The main difference is that when constructing a CDI$_r$ graph, we are not relying on observed data but using the information from the structure of the CAI or GAI representation. Our algorithm for converting CAI decompositions runs in linear time. However, our algorithm for converting GAI decompositions does not run in polynomial time (or at least, the algorithm cannot guarantee an optimal solution in polynomial time). A possible direction for future work would be to combine our algorithm with existing algorithms for creating Bayesian networks.

Converting CDI$_r$ to CAI

There are cases where CAI representations are exponentially more complex than the corresponding CDI$_r$ representation. For example, the CDI$_r$ graph in Figure 5.8 has a complexity of $O(m^3 + (n-1)m^2 + m)$. However, since there is a cycle through all attributes, the corresponding CAI representation would have a complexity of $O(m^n)$.

Converting GAI to CUI

As mentioned, Engel and Wellman show that CUI gives a weakly more compact representation of u than CAI. We next show that GAI can give an exponentially more compact representation than CUI.

We consider the GAI decomposition over $\mathbb{A} = \{A_1, A_2, A_3\}$,

$$u_{\{A_1,A_2\}}(x) + u_{\{A_2,A_3\}}(x) + u_{\{A_3,A_1\}}(x). \tag{5.12}$$

We will show that there exist utility functions of the form in Equation 5.12 for which the only CUI decomposition is the trivial one. The CUI decomposition of u, given in Equation 2.34 is

$$u(A_1, A_2, A_3) = f(A_1, A_2) + g(A_1, A_2)h(A_2, A_3). \tag{5.13}$$

(By symmetry in Equation 5.12, the order of the variables in Equation 5.13 does not matter. The actual definition of CUI given in Equation 2.34 has $h(A_2, A_3) = u(A_1 = a_1, A_2, A_3)$.

For simplicity, we consider the more general case.) For now, we ignore the $f(A_1, A_2)$ parameter in Equation 5.13. If a non-trivial CUI decomposition of Equation 5.12 exists, then

$$\begin{aligned} g(A_1, A_2)h(A_2, A_3) &= u_{\{A_1,A_2\}}(x) + u_{\{A_2,A_3\}}(x) + u_{\{A_3,A_1\}}(x) \\ g(A_1, A_2) &= \frac{u_{\{A_1,A_2\}}(x) + u_{\{A_2,A_3\}}(x) + u_{\{A_3,A_1\}}(x)}{h(A_2, A_3)} \end{aligned} \tag{5.14}$$

Equation 5.14 must hold for every $a_3 \in A_3$. Therefore,

$$\begin{aligned} &\frac{u_{\{A_1,A_2\}}(x) + u_{\{A_2,A_3\}}(x|_{A_3=a_3}) + u_{\{A_3,A_1\}}(x|_{A_3=a_3})}{h(A_2, a_3)} \\ &= \frac{u_{\{A_1,A_2\}}(x) + u_{\{A_2,A_3\}}(x|_{A_3=a'_3}) + u_{\{A_3,A_1\}}(x|_{A_3=a'_3})}{h(A_2, a'_3)}, \end{aligned} \tag{5.15}$$

for every pair of local outcomes $\{a_3, a'_3\} \subseteq A_3$. Note that Equation 5.15 is a linear constraint with two unknowns, $h(A_2, a_3)$ and $h(A_2, a'_3)$. If $|A_3| = 2$ then we have 2 unknowns in total and 1 linear constraint. However, if $|A_3| = 4$ we have 4 unknowns and 6 constraints (*i.e.* 4 choose 2). This is known as an overdetermined system, and with the exception of very rare cases, such systems do not have a solution [37]. An example of an overdetermined system is trying to fit a series of many points to a single line. Even if we include $f(A_1, A_2)$ as an additional unknown, since $f(A_1, A_2)$ is not dependent on A_3, this would only give one additional unknown. As a result, as the number of local outcomes in A_3 grows, the number of constraints grows quadratically while the number of unknowns only grows linearly. Thus, in general the nontrivial CUI decomposition in Equation 5.13 will not exist.

We can generalize Equation 5.12 to include as many attributes as we want.

Comparing CUI and CDI$_r$

Since Equation 5.12 is also a valid CDI$_r$ decomposition, in the worst-case CDI$_r$ decompositions are exponentially more compact than CUI decompositions. However, since GAI is always as compact as CDI$_r$, it is also possible for CUI decompositions to be exponentially more compact than CDI$_r$ decompositions.

Summary of Relative Complexities

To summarize our findings in this section (as well as previous findings), we use the graph shown in Figure 5.9. Each of the nodes in Figure 5.9 represents a type of independence

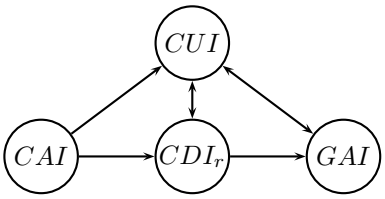


Figure 5.9: *A graphical summary of the complexity results in Section 5.2.*

model. A one-way directed edge from A to B means that B is always as compact as A and may be exponentially more compact. This property is transitive.

5.3 Experimental Results

In Section 5.2 we showed that, in the worst case, converting from GAI to CDI_r can result in an exponentially more complex representation. In this section, we investigate what happens in average cases. The average case is important because CDI_r representations are conceptually simpler than GAI representations. Hence, doing elicitation based on a CDI_r representation may be more efficient. Any increase in complexity resulting from using a CDI_r representation, however, may make the process impractical. For example, suppose we have a complexity of m^2 with a GAI representation and m^3 with a CDI_r representation. Then if $m = 10$, with the GAI representation we need to query the user about 100 outcomes, a large but feasible number of outcomes, while,with the CDI_r representation, we would have 1000 outcomes, an infeasible number of outcomes to query a user about.

5.3.1 Experimental Setup

We generated random GAI representations over 25 attributes (each with m local outcomes), converted them into CDI_r representations, and measured the resulting increase in complexity. The complexities of the GAI and CDI_r representations are given by c_{GAI} and c_{CDI_r}, respectively. For example, if $c_{GAI} = 3$, then the complexity of the GAI representation is $\Theta(m^3)$. We varied both the complexity of the GAI representation (by varying the complexity of each individual LVF) and the number of LVFs.

The challenge is that the change in complexity is dependent on how good we are at optimizing the CDI_r representation. Finding the optimal CDI_r representation can be a

hard problem [11]. While this means we could argue that, at best, it is hard to find a CDI_r representation that is not more complex than the GAI decompositions, we would like a stronger result. Our goal is to give a lower bound on the increase in complexity, regardless of how well the CDI_r representation is optimized.

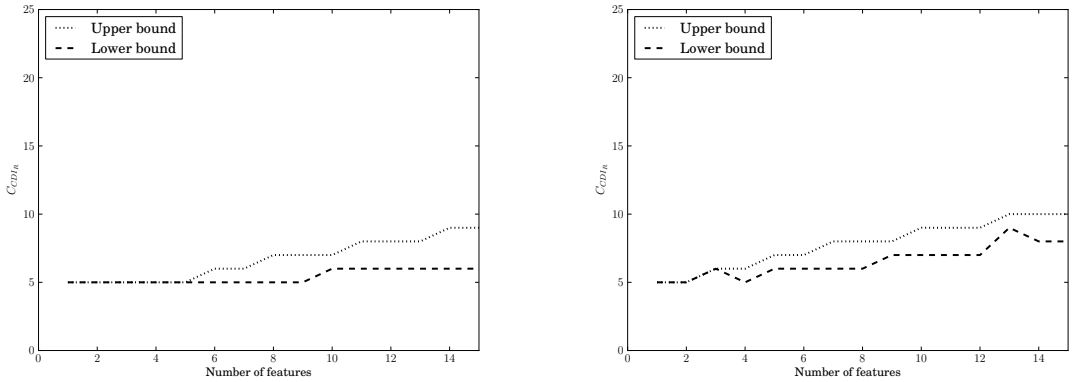
To give a lower bound on the complexity of all possible CDI_r representations, we focus on the size of the maximum clique in the undirected CDI_r representations. If we can show that a certain clique of attributes exists in all possible undirected CDI_r representations, then the minimum possible complexity for using a CDI_r representation is at least equal to the size of that clique. The actual complexity may be larger but the maximum clique size provides a lower bound. Finding the maximum clique in a graph is NP-complete [35]. We used the Bron-Kerbosch algorithm as it has been shown to be one of the fastest maximum clique finding algorithms in practice [16, 18, 38]. To find an upper bound on the complexity, we used Algorithm 3. Since ties are broken at random, for each conversion, we ran Algorithm 3 1000 times.

5.3.2 Results

We started with a complexity of $c_{GAI} = 5$ and varied the number of feaures between 1 and 15. The median and maximum complexity of the resulting CDI_r representation are shown in Figure 5.10. The median complexity grows gradually. With 15 features, the median complexity is between $\Omega(m^6)$ and $O(m^9)$. The worst-case complexity grows more quickly; with 15 features the maximum complexity is between $\Omega(m^9)$ d $O(m^{10})$. Since any increase in complexity would make CDI_r representations unusable for eliciting preferences, at best we could use CDI_r representations with up to about 9 features before the complexity starts to increase. Brazinuas and Boutilier were able to calculate minimax regret with GAI representations with 13 features [13]. In the best case with the lower bound on the median complexity, CDI_r could be used as an alternative representation. However, even with the lower bound on the maximum complexity, calculating minimax regret would be infeasible.

We were also interested in how accurate our lower and upper bounds on the complexity of the CDI_r decompositions were. Figure 5.11 shows the median and maximum difference between the lower and upper bounds as a function of the number of LVFs. With up to 6 features, on average, there is no difference between the upper and lower bounds. Even with 15 features, the median difference is only 2 orders of magnitude, *e.g.* we could have a lower bound of $\Omega(m^7)$ and an upper bound of $O(m^9)$. This gives us a fairly accurate range for the complexity. However, in the worst case, the difference is 4 orders of magnitude.

We next experimented with GAI representations with complexity $\Theta(m^{10})$, *i.e.* repre-



(a) Median complexity

(b) Maximum complexity

Figure 5.10: *The median and maximum complexity of a CDI_r representation of a utility function originally given in a GAI representation with a complexity of $\Theta(m^5)$. For example, if the upper bound on c_{CDI_r} is 6, then the complexity of the CDI_r representation is $O(m^6)$. We varied the number of features in the GAI representation between 1 and 15, each with a size of 5 attributes.*

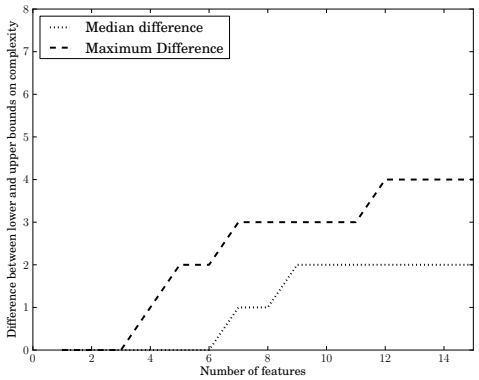
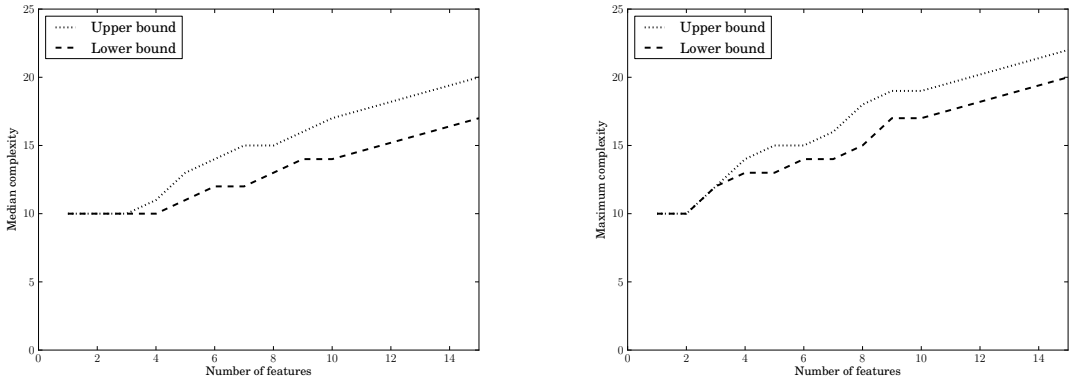


Figure 5.11: *The difference between our upper bound and lower bound on the complexity of using a CDI_r representation. For example, if difference is 2 and the lower bound on the complexity is $\Omega(m^6)$, then our upper bound is $O(m^8)$.*



(a) Median complexity

(b) Maximum complexity

Figure 5.12: *The median and maximum complexity of a CDI$_r$ representation of a utility function originally given in a GAI representation with a complexity of $\Theta(m^{10})$. We varied the number of LVFs in the GAI representation between 1 and 15, each with a size of 10 attributes.*

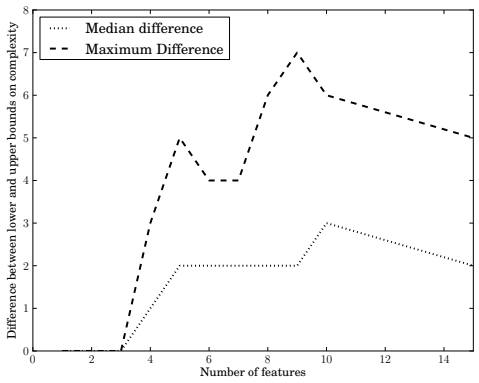


Figure 5.13: *The difference between our upper bound and lower bound on the complexity of using a CDI$_r$ representation. The original GAI representations had a complexity of $\Theta(m^{10})$. The spike is due to the fact that the upper bound must stop increasing before the lower bound stops increasing since the worst-case upper bound is $O(m^{25})$.*

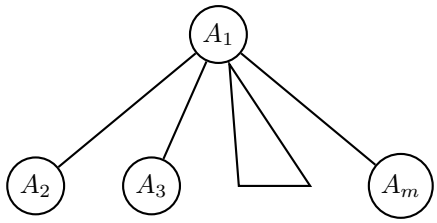


Figure 5.14: *The skeleton CDI$_r$ graph of the LVFs in Equation 5.16.*

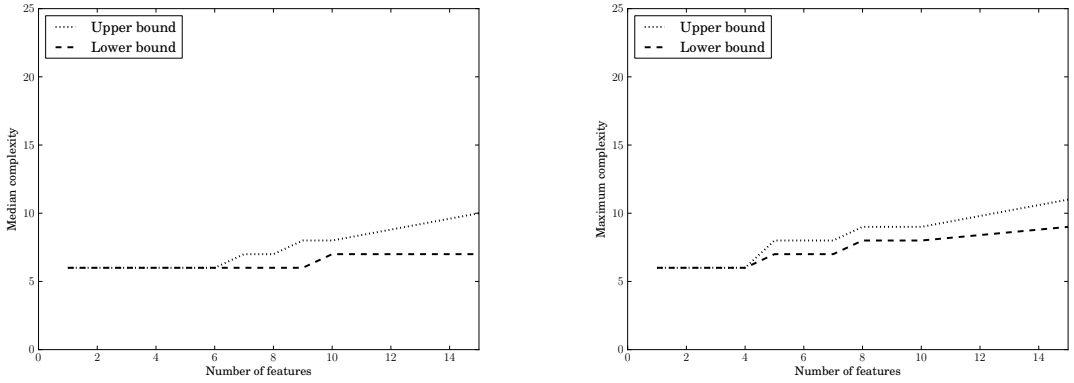
sentations where each feature was composed of 10 attributes. Since GAI representations with a complexity of $\Theta(m^{10})$ are not practical to deal with, these experiments had more of a theoretical motivation. The median and maximum complexity of the resulting CDI$_r$ representations are shown in Figure 5.12. The median complexity increases faster than the results in Figure 5.10(a), and with 15 features, we have a range of complexity between $\Omega(m^{17})$ and $O(m^{20})$. The maximum complexity also increases faster than in Figure 5.12(b) and with 15 LVFs we have a worst-case range of complexity of $\Omega(m^{20})$ and $O(m^{22})$.

Once again, we also compared the difference between the lower and upper bounds on the complexity. The results are shown in Figure 5.13. The median difference between lower and upper bounds is relatively small; at most either 2 or 3 orders of magnitude. This is comparable to the median difference shown in Figure 5.11 for 5 attributes per LVF. However, the maximum difference, 7 orders of magnitude is considerable and much larger than the maximum difference seen in Figure 5.11. We see that with enough LVFs, the difference between our lower and upper bounds starts to decrease. This is because the upper bound must stop increasing before the lower bound stops increasing.

We next investigated "hard" conversion cases where we try to maximize the difference between the lower bound and upper bound of the complexity of the CDI$_r$ decomposition. With our current approach, there is no guarantee that the resulting undirected CDI$_r$ graph will be *connected*. In a connected graph, there is a path between any two nodes. One possible way to guarantee a connected graph is to add the LVFs

$$v_{A_1,A_2}(x), v_{A_1,A_2}(x), \ldots, v_{A_1,A_m}(x), \tag{5.16}$$

to all of the GAI representations. The undirected CDI$_r$ graph resulting from just these LVFs is shown in Figure 5.14. Note that as long as the complexity of the GAI representation is at least $\Theta(m^2)$, the addition of the LVFs in Equation 5.16 will not increase the complexity of the GAI representation. Thus, these LVFs will increase the number of edges in our undirected CDI$_r$ graph (thus hopefully increasing the complexity of the CDI$_r$ representation) without increasing the complexity of the GAI representation. The median



(a) Median complexity

(b) Maximum complexity

Figure 5.15: *The median and maximum complexity of a CDI$_r$ representation of a utility function originally given in a GAI representation with a complexity of $\Theta(m^5)$. We varied the number of LVFs in the GAI representation between 1 and 15, each with a size of 5 attributes. All of the GAI representations included the LVFs in Equation 5.16.*

and worst case complexity of the resulting CDI$_r$ representations are shown in Figure 5.15. These results can be compared to the results in Figure 5.10. We see that for the lower and upper bounds for both the median and maximum complexity, there is an increase of one order of magnitude.

To create harder conversion cases, we need to find a way of minimizing the size of the maximum clique (since our lower bound is determined by the size of the maximum clique) while maximizing the number of edges in the CDI$_r$ (since more edges usually results in a higher complexity.) For a given number of attributes n and a clique size $q+1$, the maximum number of edges a graph can have without having a clique of size $q+1$ is $t(n,q)$ where [58]

$$t(n,q) \leq \frac{(q-1)n^2}{2q}.$$

Furthermore, the graph containing $t(n,q)$ edges without a clique of size $q+1$ is unique up to isomorphism and is known as the *Turán graph* [58]. The Turán graph can easily be constructed by having an edge between attributes A_i and A_j if and only if i and j are incongruent modulo q, that is $(i-j) \mod q \neq 0$ [21].

We created Turán graphs with 25 attributes and varied the maximum clique size be-

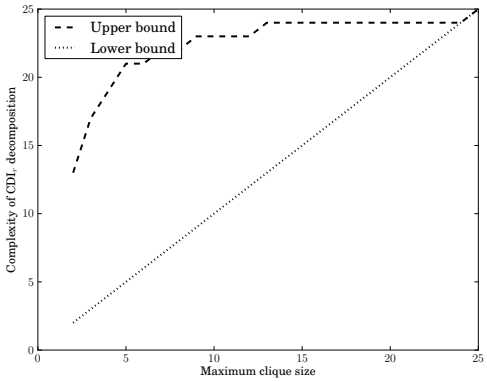


Figure 5.16: *The maximum complexity of a CDI$_r$ representation based on a Turán graph.*

tween 2 and 25. The maximum clique in the (n, q) Turán graph is of size of q which allows us to easily give a lower bound on c_{CDI_r}. The lower and upper bounds on c_{CDI_r} are shown in Figure 5.16. As expected, the upper bounds are considerably larger than the lower bounds. For example, with a maximum clique size of only 5, we have an upper bound of $O(m^{21})$.

These results show that for simple utility functions, *e.g.* utility functions composed of only a small number of LVFs with each LVF having a low complexity, CDI$_r$ decompositions may be an efficient alternative to GAI decompositions. As either the number of LVFs increases or the complexity of the LVFs increases, however, the complexity of the CDI$_r$ decompositions can be noticeably larger than that of the equivalent GAI decompositions.

Experimental Conclusion

Braziunas and Boutilier were able to solve minimax regret calculations using GAI decompositions with up to 13 LVFs (or 13 features) with up to 5 attributes in each LVF. In our experiments, the best-case complexity for CDI$_r$ with 13 LVFs with 5 attributes each was between $\Omega(m^6)$ and $O(m^8)$. Even in simple scenarios, with 15 features each with 5 attributes, the worst case complexity was between $\Omega(m^9)$ and $O(m^{10})$. None of these values would result in a representation suitable to use for querying a user about. Only in the best of these cases, *e.g.* $O(m^6)$ or $O(m^7)$, would we have representations suitable for minimax regret calculations.

Our results showed that, on average, we were able to find accurate lower and upper

bounds on the complexity of CDI_r representations. The average range of complexities was usually at most two or three orders of magnitude. We were also able to find many cases where we found the exact complexity, that is, the lower complexity bound and the upper complexity bound were equal. This shows that our algorithm for determining upper bounds on complexity, despite being simple and random, can be effective. We also investigated many cases where there was a considerable difference between the lower and upper bounds. These cases highlight areas for future research.

Since our results were based on random cases, our results may not apply to specific cases of interest. There may be real life examples where CDI representations are equally as compact as GAI representations. Our methods for determining a lower and upper bound on the complexity of the conversion can be applied to any set of CDI representations. As a result, given a specific real world scenario, we can easily check whether a CDI representation would be useful.

5.4 Conclusion

While many different multiattribute utility independence (MUI) models have been proposed, we feel there has been little work done on comparing the relative compactness of these different models. While the proposed MUI models were all designed to investigate interesting properties in multiattribute preferences, a model is not worthwhile if it cannot compactly represent utility preferences. Our main proofs show that GAI can be exponentially more compact than CDI_r, CDI_r can be exponentially more compact than CAI and that both GAI and CDI_r can be exponentially more compact than CUI.

The worst case is often an extreme overestimate of the average case. Therefore, it is important to investigate the average relative compactness of these models. In the second half of this chapter, we presented experimental results which showed that, with the exception of relatively simple cases, GAI is often noticeably more compact than CDI_r even in average cases.

Chapter 6

Conclusion

In order for computers to help people make good decisions, we must have good models of peoples' preferences. The major theme in this thesis was to look at ways of making preference elicitation more practical. We accomplished this goal in two distinct ways. First, we examined the underlying models that are traditionally assumed in preference elicitation and found ways to make these models more realistic. Secondly, we examined ways of making the elicitation process more efficient. We improved efficiency both by introducing new ways of measuring and estimating regret and by comparing different models for multiattribute preferences. The comparison of multiattribute models is important since there are so many to choose from; previously there had been few comparisons made between differnt models. Our comparisons provide a benchmark by which to judge different models.

6.1 Contributions

The main contributions of this thesis are:

- We demonstrated that *standard gamble queries*, the standard method for eliciting preferences, are not compatible with *cumulative prospect theory* (CPT), the standard descriptive model of human reasoning under uncertainty.
- The *Gamble Equivalence Method*, a new querying method which is compatible with CPT and can be used to efficiently reduce the minimax regret.
- *Learnt Probabilistic Minimax Regret* (lPrMMR) and *hypothesis-based Probabilistic Minimax Regret* (hPrMMR), two new methods for estimating and bounding regret.

Unlike *minimax regret* (MMR), lPrMMR and hPrMMR are able to provide tight upper bounds of the actual regret. The bounds provided by lPrMMR and hPrMMR are probabilistic, that is, there is some probability that the actual regret is higher than the given upper bound; however, the controller (the person using the preference elicitation) is able to choose the probability of error. The lower the probability of error, the higher the upper bound. If the controller decides on a probability of error equal to 0, both lPrMMR and hPrMMR behave similarly to MMR. As the probability of error increases, lPrMMR and hPrMMR start behaving similarly to *expected regret*. However, lPrMMR and hPrMMR do not need to make the strong prior knowledge assumptions made by expected regret. Both lPrMMR and hPrMMR are especially well suited to eliciting the preferences from a sequence of users.

- Finally, we showed that with the *multiattribute utility independence* (MUI) model *common difference independence* (CDI_r), we can calculate MMR polynomially with respect to the size of the CDI_r model. We also showed, however that in the worst case, utility representations using CDI_r will be exponentially more complex than with GAI. We also showed that, even in the average case, GAI provides a significantly more compact representation.

6.2 Future Work

We review several different possibilities for future work.

6.2.1 Human Experiments

We would like to make use of our work in human experiments. There has been some work done with human experiments and AI preference elicitation [15]. We are unaware of any AI work done which involved uncertainty or risk. Most of the work done with human experiments and risk has been in management science research [2, 5, 7, 61, 65, 71].

Of the many implementational issues to consider, the most important is peoples' ability to do calculations. All of our work has assumed that people are able to perfectly calculate and articulate utility values and probability weightings. This is not the case in practice, which results in people giving inconsistent responses to queries. If we can bound the inconsistencies, we can incorporate them into our model. By limiting the accuracy of the utility constraint sets, though the inconsistencies will also limit how low we can drive the minimax regret. Thus, instead of querying until we get the minimax regret below a given

threshold, we might only be able to query until the maximum utility gap is of a certain size, and say that we cannot improve on the resulting minimax regret. Recent work has examined possible methods for dealing with the lack of precision in preferences [47].

6.2.2 Eliciting Intertemporal Preferences

It is easy to generalize many of applications for preference elicitation to intertemporal settings where utilities are dependent on time. For example, would you rather pay extra to avoid long layovers before the holiday or after the holiday? Postponing running the dryer until the middle of the night might save money, but you might need clothing right now.

There has been considerable work done in management science and behavioural economics on eliciting preferences in intertemporal settings [3, 4]. However, to our knowledge, there has been little work done in AI on eliciting preferences in intertemporal settings. The closest we are aware of is eliciting preferences for constructing a Markov Decision Process (MDP) [53].

As with preferences under uncertainty, there is a standard economically rational model of how people reason about intertemporal preferences [54]. This model is known as *discounted utility*. For example, MDPs are built on the basis of discounted utility [54]. However, there is strong experimental evidence that people do not actually reason using discounted utility [8, 40]. This evidence means that discounted utility is not a reasonable descriptive model. To make matters worse, there is a strong feeling that discounted utility is not even a reasonable prescriptive model [40].

This means that there are many problems with generalizing AI preference elicitation to an intertemporal setting. How do we generalize SGQs or outcome queries? How do we generalize minimax regret? What are good descriptive and prescriptive models to use? Intertemporal preferences have strong connections with multiattribute preferences which should be explored.

6.2.3 Improving the Performance of Learnt Probabilistic Regret

In the limit, as we process more users, learnt probabilistic regret will be able to give accurate upper bounds on the actual regret. For example, if we choose a probability of error equal to 0.2, then learnt probabilistic regret should give us a value that is an upper bound 80% of the time. As we saw in the experimental results for learnt probabilistic regret, even after processing hundreds of users, we were not close to this level of accuracy.

There are only a limited number of ways to improve the performance of lPrMMR which are able to keep the theoretical properties of lPrMMR. Massart showed that the constant in the Dvoretzky-Kiefer-Wolfowitz inequality cannot be improved [42]. This means that the only ways to improve performance are by processing more users or by requiring a minimum level of accuracy for the utility constraints for each user. Requiring a minimum level of accuracy can be self-defeating. Aminimum level of accuracy requires a minimum number of queries, which is at odds with the goal of minimizing the number of queries.

Other methods for improving the performance would forfeit the theoretical properties of lPrMMR, but could work well in practice. For example, in Figure 4.3, we mentioned that we consider a set of utility constraints C to be compatible with the constraints C' if and only if $C \subseteq C'$. It might be worthwhile to include any set which overlaps with C'. For example, if C and C' overlap, we could consider the set $C \cap C'$ and weight C by

$$\frac{||C \cap C'||}{||C'||},$$

i.e. the percentage of C that is compatible with C'. Another possible heuristic would be to ignore some outcomes. As the number of outcomes increases, it becomes easier for utility constraint sets to become incompatible. By focusing on only a few important outcomes, *e.g.* those outcomes which happen often or have a large utility associated with them, we increase the number of compatible utility constraint sets. However, by ignoring some outcomes, we are also ignoring possible correlations between utility values which could skew our results. It might also be possible to improve performance by accounting for correlations between probability distributions.

6.2.4 Examining Probabilistic Regret from a Probably Approximately Correct Perspective

We would also like to examine how probabilistic regret is related to Probably Approximately Correct (PAC) learning. The focus would on understanding the relationship between the Dvoretzky-Kiefer-Wolfowitz inequality and PAC learning.

6.2.5 Expanding our Comparison of Multiattribute Utility Independence Models

Our comparison of multiattribute utility independence (MUI) models focused on models which could easily be used to compute minimax regret. There is an obvious benefit to

comparing additional MUI models and examining if any of them also allow for easy computation of minimax regret [45]. It would be especially beneficial to continue the work done on comparing compactness in average cases, not just the worst case.

There is considerable use of MUI models in management science research [1, 24, 67]. Examining this research would give us a better idea of the real-world uses of MUI models. This would allow us to compare the relative compactness of MUI models for real-world settings.

6.2.6 Improving the Efficiency of Multiattribute Preference Elicitation

Responding to standard gamble queries is not the only way for users to express preferences. Recent work has examined new methods for reasoning about quantative preferences [43]. Some of these methods may be especially well suited for specific MUI models. Incorporating these methods could be one method for improving the elicitation process.

References

[1] Ali E. Abbas. Invariant utility functions and certain equivalent transformations. *Decision Analysis*, 4:17–31, 2007. 143

[2] Mohammed Abdellaoui. Parameter-free elicitation of utility and probability weighting functions. *Management Science*, 46:1497–1512, 2000. 40, 140

[3] Mohammed Abdellaoui, Arthur E. Attema, and Han Bleichrodt. Intertemporal tradeoffs for gains and losses: An experimental measurement of discounted utility. *The Economic Journal*, 120:845–866, 2010. 141

[4] Mohammed Abdellaoui, Enrico Diecidue, and Ayse Onculer. Risk preferences at different time periods: An experimental investigation. *Management Science*, 57:975–987, 2011. 141

[5] Mohammed Abdellaoui, Frank Vossmann, and Martin Weber. Choice-based elicitation and decomposition of decision weights for gains and losses under uncertainty. *Management Science*, 51:1384–1399, 2005. 40, 140

[6] M. Allais. Le comportement de l'homme rationnel devant le risque, critique des postulats et axiomes de l'école Américaine. *Econometrica*, 21:503–546, 1953. 10

[7] Han Bleichrodt and Jose Luis Pinto. A parameter-free elicitation of the probability weighting function in medical decision analysis. *Management Science*, 46:1485–1496, 2000. 39, 140

[8] Han Bleichrodt, Kirsten I.M. Rohde, and Peter P. Wakker. Non-hyperbolic time inconsistency. *Games and Economic Behavior*, 66:27–38, 2009. 141

[9] Craig Boutilier. On the foundations of expected expected utility. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence*, Acapulco, 2003. 16

[10] Craig Boutilier, Relu Patrascu, Pascal Poupart, and Dale Schuurmans. Constraint-based optimization and utility elicitation using the minimax decision criterion. *Artificial Intelligence*, 170:686–713, 2006. 2, 3, 17, 19, 20, 21, 51, 119

[11] Ronen I. Brafman and Yagil Engel. Decomposed utility functions and graphical models for reasoning about preferences. In *Proceedings of the Twenty-Second Conference on Artificial Intelligence*, 2010. 2, 4, 26, 30, 31, 114, 129, 132

[12] Darius Braziunas and Craig Boutilier. Preference elicitation and generalized additive utility. In *Proceedings of the Twenty-First Conference on Artificial Intelligence (AAAI-06)*, 2006. 30

[13] Darius Braziunas and Craig Boutilier. Minimax regret based elicitation for generalized additive utilities. In *Proceedings of the Twenty-Second Conference on Artificial Intelligence*, 2007. 3, 31, 119, 132

[14] Darius Braziunas and Craig Boutilier. Elicitation of factored utilities. *AI Magazine*, 29, 2008. 2

[15] Darius Braziunas and Craig Boutilier. Assessing regret-based preference elicitation with the utpref recommendation system. In *Proceedings of the 11th ACM conference on Electronic Commerce (EC-11)*, EC '10, pages 219–228, New York, NY, USA, 2010. ACM. 140

[16] Coen Bron and Joep Kerbosch. Algorithm 457: finding all cliques of an undirected graph. *Commun. ACM*, 16:575–577, September 1973. 132

[17] Colin F. Camerer, George Loewenstein, and Matthew Rabin, editors. *Advances in Behavorial Economics*. Princeton University Press, 2003. 49

[18] Frederic Cazals and Chinmay Karande. A note on the problem of reporting maximal cliques. *Theoretical Computer Science*, 407:564–568, 2008. 132

[19] Urszula Chajewska and Daphne Koller. Utilities as random variables: Density estimation and structure discovery. In *Proceedings of the Sixteenth Annual Conference on Uncertainty in Artificial Intelligence (UAI-00)*, 2000. 15, 69

[20] Urszula Chajewska, Daphne Koller, and Ron Parr. Making rational decisions using adaptive utility elicitation. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 363–369, Austin, TX, 2000. 1, 2, 14, 15, 34, 69

[21] Chong-Yun Chao and George A. Novacky. On maximally saturated graphs. *Discrete Mathematics*, 41:139–143, 1982. 136

[22] Li Chen and Pearl Pu. Survey of preference elicitation methods. Technical report, Ecole Politechnique Federale de Lausanne, 2004. 1, 14

[23] Robin P. Cubitt and Daniel Read. Can intertemporal choice experiments elicit time preferences for consumption? *Experimental Economics*, 10:369–389, 2007. 49

[24] James S. Dyer, Peter C. Fishburn, Ralph E. Steuer, Jyrki Wallenius, and Stanley Zionts. Multiple criteria decision making, multiattribute utility theory: The next ten years. *Management Science*, 38:645–654, 1992. 143

[25] Yagil Engel and Michael P. Wellman. CUI networks: A graphical representation for conditional utility independence. *Journal of Artificial Intelligence Research*, 31:83–112, 2008. 2, 4, 26, 27, 30, 32

[26] Peter H. Farquhar. Utility assessment methods. *Management Science*, 30:1283–1300, 1984. 38

[27] Peter C. Fishburn. Interdependence and additivity in multivariate, unidimensional expected utility theory. *International Economic Review*, 8:335–342, 1967. 2, 3, 26, 29

[28] Philip A. Haile and Elie Tamer. Inference with an incomplete model of english auctions. *Journal of Political Economy*, 111(1):pp. 1–51, 2003. 94

[29] Greg Hines and Kate Larson. Preference elicitation for risky prospects. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems (AAMAS-10)*, Toronto, Canada, 2010. 51

[30] Greg Hines and Kate Larson. Efficiently eliciting preferences from a group of users. In *Proceedings of the 2011 AAAI Workshop on Interactive Decision Theory and Game Theory*, 2011. 1

[31] IDEAS. Ideas rankings. http://ideas.repec.org/top/, 2011. 10

[32] Daniel Kahneman, Jack L. Knetsch, and Richard H. Thaler. Experimental test of the endowment effect and the coase theorem. *Journal of Political Economy*, 98:1325–1348, 1990. 10

[33] Daniel Kahneman and Amos Tversky. Prospect theory: An analysis of decision under risk. *Econometrica*, 47:263–291, 1979. 10

[34] Daniel Kahneman and Amos Tversky. Choices, values and frames. *American Psychologist*, 39:341–350, 1984. 11

[35] Richard Karp. Reducibility among combinatorial problems. In Raymond E Miller and James W Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press, NY, 1972. 132

[36] Ralph Keeney and Howard Raiffa. *Decisions with multiple objectives: Preferences and value tradeoffs*. Wiley, New York, 1976. 2, 14, 26, 27

[37] Bernard Kolman and David R. Hill. *Elementary Linear Algebra*. Prentice Hall, 2000. 130

[38] Oleksii Kuchaiev. Source code for Bron Kerbosch algorithm. http://www.kuchaev.com/projects.html. 132

[39] Raul H.C. Lopes, Ivan Reid, and Peter R. Hubson. The two-dimensional kolmogorov-smirnov test. In *Proceedings of the Eleventh International Workshop on Advanced Computing and Analysis Techniques in Physics Research*, 2007. 84

[40] George Lowenstein and Drazen Prelec. Anomalies in intertemporal choice: Evidence and an interpretation. *Quarterly Journal of Economics*, 107:573–597, 1992. 141

[41] Andreu Mas-Colell, Michael D. Whinston, and Jerry R. Green. *Microeconomic Theory*. Oxford University Press, 1995. 5, 6, 7, 8, 9, 42, 60

[42] Pascal Massart. The tight constant in the Dvoretzky-Kiefer-Wolfowitz inequality. *The Annals of Probability*, 18(3):pp. 1269–1283, 1990. 142

[43] Michael McGeachie and Jon Doyle. The local geometry of multiattribute tradeoff preferences. *Artificial Intelligence*, 175:1122–1152, 2011. 143

[44] Tom Mitchell. *Machine Learning*. McGraw-Hill, 1997. 72, 84, 91

[45] Piero La Mura and Yoav Shoham. Expected utility networks. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI)*, 1999. 143

[46] Balas Kausik Natarajan. *Machine Learning , A Theoretical Approach*. Morgan Kaufmann Publishers, 1991. 93

[47] Meltem Ozturk, Marc Pirlot, and Alexis Tsoukias. Representing preferences using intervals. *Artificial Intelligence*, 175:1194–1222, 2011. 141

[48] John W. Pratt and Jean D. Gibbons. *Concepts of Nonparametric Theory*. Springer-Verlag, 1981. 85, 91

[49] Drazen Prelec. The probability weighting function. 1995. 38

[50] Matthew Rabin. Risk aversion and expected-utility theory: A calibration theorem. *Econometrica*, 68:1281–1292, 2000. 54

[51] George Rebane and Judea Pearl. The recovery of causal poly-trees from statistical data. In *Proceedings of the 3rd Workshop on Uncertainity in Artificial Intelligence*, 1987. 126

[52] Kevin Regan and Craig Boutilier. Regret-based reward elicitation for Markov decision processes. In *Proceedings of the 25th International Conference on Uncertainty in Artificial Intelligence (UAI-09)*, Montreal, Canada, 2009. 1, 14

[53] Kevin Regan and Craig Boutilier. Robust online optimization of reward-uncertain mdps. In *Proceedings of the Twenty Second International Joint Conference on Artificial Intelligence (IJCAI-11)*, 2011. 141

[54] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 2nd edition, 2003. 2, 5, 6, 8, 9, 34, 141

[55] Chris Starmer. Developments in non-expected utility theory: The hunt for a descriptive theory of choice under risk. *Journal of Economic Literature*, 38:332–382, 2000. 2, 9, 10, 34

[56] James Stewart. *Calculus*. Brooks/Cole Publishing Company, 1999. 45

[57] Nassim Nicholas Taleb. *The Black Swan*. Random House, 2007. 75

[58] Paul Turan. On an extremal problem in graph theory. *Matematiko Fizicki Lapok*, 48:436–452, 1941. 136

[59] Amos Tversky and Daniel Kahneman. Advances in prospect theory: Cumulative representation of uncertainty. *Journal of Risk and Uncertainty*, 5(4):297–323, 1992. 10, 12, 36, 97, 107

[60] Leslie Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, November 1984. 93

[61] Gijs van de Kuilen and Peter P. Wakker. The midweight method to measure attitudes toward risk and ambiguity. *Management Science*, 57:582–598, 2011. 41, 140

[62] Vladimir Vapnik. *The Nature of Statistical Learning Theory*. Springer, 2nd edition, 1999. 94

[63] John von Neumann and Oskar Morgenstein. *Theory of games and economic behavior*. Princeton University Press, 1947. 8, 9

[64] Perukrishnen Vytelingum, Sarvapali D. Ramchurn, Thomas D. Voice, Alex Rogers, and Nicholas R. Jennings. Trading agents for the smart electricity grid. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: (AAMAS-10)*, AAMAS '10, pages 897–904, Richland, SC, 2010. International Foundation for Autonomous Agents and Multiagent Systems. 1

[65] Peter Wakker and Daniel Deneffe. Eliciting von Neumann-Morgenstern utilities when probabilities are distorted or unknown. *Management Science*, 42:1131–1150, 1996. 39, 40, 42, 140

[66] Peter Wakker, Ido Erev, and Elke U. Weber. Comonotonic independence: The critical test between classical and rank-dependent utilities theories. *Journal of Risk and Uncertainty*, 9:195–230, 1994. 13

[67] Jyrki Wallenius, James S. Dyer, Peter C. Fishburn, Ralph E. Steuer, Stanley Zionts, and Kalyanmoy Deb. Multiple criteria decision making, multiattribute utility theory: Recent accomplishments and what lies ahead. *Manage. Sci.*, 54:1336–1349, July 2008. 143

[68] Tianhan Wang and Craig Boutilier. Incremental utility elicitation with the minimax regret decision criterion. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI-03)*, pages 309–318, Acapulco, Mexico, 2003. 1, 2, 14, 17, 19, 22, 24, 25, 34, 53

[69] Larry Wasserman. *All of Statistics*. Springer, 2004. 69, 86, 91, 108

[70] Larry Wasserman. *All of Nonparametric Statistics*. Springer, 2006. 77

[71] George Wu and Richard Gonzalez. Curvature of the probability weighting function. *Management Science*, 42:1676–1690, 1996. 11, 36, 38, 39, 43, 45, 53, 140