Inferring Noncompensatory Choice Heuristics

by

Michael J Yee

BS, Gordon College (2000)
SM, Massachusetts Institute of Technology (2003)

Submitted to the Sloan School of Management
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy in Operations Research

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2006

© Massachusetts Institute of Technology 2006. All rights reserved.

Author .....
Sloan School of Management
May 18, 2006

Certified by .....
James B. Orlin
Edward Pennell Brooks Professor of Operations Research
Thesis Supervisor

Accepted by .....
Dimitris J. Bertsimas
Boeing Professor of Operations Research
Co-director, Operations Research Center

MASSACHUSETTS INSTITUTE OF TECHNOLOGY
JUL 2 4 2006
LIBRARIES

ARCHIVES

Inferring Noncompensatory Choice Heuristics

by

Michael J Yee

Submitted to the Sloan School of Management
on May 18, 2006, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy in Operations Research

Abstract

Human decision making is a topic of great interest to marketers, psychologists, economists, and others. People are often modeled as rational utility maximizers with unlimited mental resources. However, due to the structure of the environment as well as cognitive limitations, people frequently use simplifying heuristics for making quick yet accurate decisions. In this research, we apply discrete optimization to infer from observed data if a person is behaving in way consistent with a choice heuristic (e.g., a noncompensatory lexicographic decision rule).

We analyze the computational complexity of several inference related problems, showing that while some are easy due to possessing a greedoid language structure, many are hard and likely do not have polynomial time solutions. For the hard problems we develop an exact dynamic programming algorithm that is robust and scalable in practice, as well as analyze several local search heuristics.

We conduct an empirical study of SmartPhone preferences and find that the behavior of many respondents can be explained by lexicographic strategies. Furthermore, we find that lexicographic decision rules predict better on holdout data than some standard compensatory models.

Finally, we look at a more general form of noncompensatory decision process in the context of consideration set formation. Specifically, we analyze the computational complexity of rule-based consideration set formation, develop solution techniques for inferring rules given observed consideration data, and apply the techniques to a real dataset.

Thesis Supervisor: James B. Orlin
Title: Edward Pennell Brooks Professor of Operations Research

Acknowledgments

This thesis would not have been possible without the help and support of others. First, I'd like to thank my thesis advisor, Jim Orlin, for his guidance and wisdom throughout the years of my research. Working with an outstanding researcher like Jim is very exciting, although a bit dangerous—his sharp mind is so quick that there's always the possibility that he might solve your problem right on the spot!

I'd also like to thank John Hauser and Ely Dahan for their significant contributions to the research. John served as a co-advisor for me and never ceased to amaze me with his broad yet deep knowledge of all aspects of marketing and business. I also benefitted greatly from his wisdom and experience as an academic who also has extensive knowledge of industry. Ely brought a passion for relevance to the techniques and helped me out tremendously with his experience in conducting empirical studies. Both Ely and John also did a wonderful job presenting this work at various conferences, successfully generating a lot of interest in our approach.

Much of this work relied heavily on computing, and I used many excellent freely available software packages and tools. Octave was used for numerical computing, GLPK was used for linear and integer programming, Java was used for general programming, PHP was used for constructing the web-based survey, and LaTeX was used for typesetting the thesis.

Last but certainly not least, I thank my wife Kara and daughter Esther for graciously tolerating the graduate student lifestyle and especially the last few hectic months! The support and prayers of my immediate and extended family were essential during these years.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

1.1 Motivation

One of humankind's most interesting characteristics is the ability to think about thinking. When reflecting on thought processes, however, it is not always clear that how we retrospectively perceive and describe them accurately matches up with the mysterious emergent behavior of the brain (coupled with other physiological factors such as emotions). Even analysis of a thought process that results in a tangible outcome (e.g., choosing an option from a set of alternatives), is not always clear cut. Nevertheless, there is great interest in studying decision making at the individual level.

Psychologists approach decision making in a variety of ways, ranging from topics like memory and cognition to framing, biases, etc. Economists are interested in decision making since ultimately higher level system behavior and dynamics arise from individual level consumer behavior. Insight into how people evaluate options and make decisions can also be helpful for studying contracts, auctions, and other forms of negotiation.

Computer scientists involved in constructing intelligent systems have often looked to human decision making as an existing example that works well in practice. For example, chess playing programs, while admittedly having an architecture significantly different from human brains, often incorportate heuristics for making the search more

intelligent. One such example is the "killer heuristic." If the computer discovers a "killer" reply to one of the computer's possible moves (i.e., a reply that makes the move unplayable), there is a good chance that it might also be a killer reply to other moves and should be checked as the first reply to those as well.

Researchers in marketing science (and consumer behavior) have perhaps the keenest interest in learning how consumers choose between products. Knowing how consumers evaluate products and product features can aid in product design and also serve to focus and improve advertising. Ultimately, market researchers desire models of decision making that have robust predictive ability.

1.2 Background

Decision making is an interesting field of study for several reasons. One is that nearly everyone makes numerous decisions everyday—some weighty and some not, some repeated, some only once. For some people, professional success directly depends on making good decisions with high frequency, e.g., effectively managing a portfolio of securities. The following anecdote illustrates an example of decision making in today's modern world.

A certain graduate student was going to be attending a conference in a month and needed to book a hotel and flight. The student's advisor, who would be financing the trip, suggested that the student try not to travel too extravagantly if possible. Armed with this objective, the student headed to the internet.

The conference was being held at a particular hotel that guaranteed a limited number of rooms at a special group rate. However, one potential way to reduce cost would be to find a cheaper hotel close by. Entering the address of the conference hotel into Yahoo.com's yellow pages and searching for nearby hotels brought up a list of about 10-15 hotels within 0.5 miles. This cutoff was chosen because the student felt that half a mile was a reasonable distance to walk (and taxi costs would be avoided).

After pricing several of the alternatives (starting with brands that the student recognized), many had exorbitant rates. A couple unknown hotels were found that

appeared very reasonable with respect to price. However, when starting to book a reservation through travel.yahoo.com, the student noticed very poor user ratings (dirty, worst hotel ever, would never stay their again, etc.). At that point, the student decided to restrict the search to hotels that had at least an average user rating of 3 or 4 stars. This drastically reduced the available options. The few hotels that were within 0.5 miles and had acceptable ratings either had no rooms available or were very expensive.

Faced with this dilemma, the student relaxed the distance preference. However, the set of options was becoming complex. Ultimately, the student enlisted the help of orbitz.com to find a list of hotels in the area that had vacancies and sorted the list by price. An alternative materialized that had a decent rating, a decent price, and was located within a mile or two from the conference. Success!

Several interesting observations can be made. First, the student looked at only a small number of cues per alternative (distance, brand/chain, price, rating). Second, no tradeoffs between cues/features were made at any point (e.g., figuring out how much each tenth of a mile was worth). Third, the importance of cues changed throughout the decision process. Some important questions arise. Was the final decision optimal? Was the decision process even rational?

Theories about decision making are intimately tied to theories about rationality. The following overview of unbounded and bounded rationality closely follows Gigerenzer and Todd [22]. Gigerenzer and Selten [21] and Chase et al [10] also provide good historical overviews.

Unbounded Rationality. Models of unbounded rationality are reminiscent of Pierre-Simon Laplace's idea that a superintelligence with complete knowledge of the state of the universe at a particular instant would be able to predict the future (which would be deterministic and certain). In a similar way, unbounded rationality does not take into account constraints of time, knowledge (information), and computational ability. The models do account for uncertainty (unlike Laplace's vision), and ultimately take the form of maximizing expected (subjective) utility. One way to characterize unbounded rationality is that it focuses on optimization. The core as-

sumption is that people should and are able to make optimal decisions (in a subjective utility sense). Thus, models of rationality have often been viewed as both descriptive and prescriptive.

Proponents of unbounded rationality sometimes acknowledge human limitations while arguing that the outcomes of decision processes are still consistent with unbounded rationality. That is, humans act as if unboundedly rational. Alternatively, a modification known as *optimization under constraints* tries to incorporate limited information search in an attempt to be more faithful to reality. In optimization under constraints, information search is stopped once the cost of the next piece of information outweights the benefits. However, adding the determination of optimal stopping rules to the decision process can require even more calculation and information gathering than plain unbounded rationality!

Bounded Rationality In the 1950s, Herbert Simon introduced the important notion of *bounded rationality* ([59, 60]). In a later paper, Simon summarized the concept well with the following metaphor: "Human rational behavior...is shaped by a scissors whose two blades are the structure of task environments and the computational capabilities of the actor" [61, p. 7]. From the perspective of bounded rationality, human decision processes are viewed as shaped by both mental limitations and the structure of the environment. Hallmarks include simple and limited information search as well as simple decision rules. For example, Simon introduced the process of satisficing in which a search is stopped once an alternative meets aspiration levels for all features.

Another term that Gigerenzer and others have used to represent the key ideas of bounded rationality is *ecological rationality.* One reason for the relabeling is that the term bounded rationality has often been misapplied, e.g., as a synonym for optimization under constraints. Furthermore, emphasis has often been placed on the limitations aspect of bounded rationality instead of the structure of the tasks. The reality is that simple heuristics and environmental structure can work together as a viable alternative to optimization.

Gigerenzer et al [23] describe simple heuristics that are ecologically rational as "fast and frugal"—fast because they do not require complex computations, and frugal

because they do not require too much information. One class of fast and frugal heuristics is one reason decision making (e.g., the "Take The Best" heuristic described in Gigerenzer and Goldstien [20]). It circumvents the problem of having to combine multiple cues, which requires converting them into a common currency. This is often difficult if not impossible, e.g., trading off living closer to family versus having a more prestigious job. Models that involve maximizing expected subjective utility require that options/features be commensurate. Thus, fast and frugal heuristics are often *noncompensatory*, while unbounded rationality models are often compensatory.

How can simple heuristics work. How can fast and frugal heuristics be so simple and yet still work? There are a few reasons. One is that they are specific to particular environments and exploit the structure. However, they are not *too* specific, i.e., they still often have much fewer parameters than more complex models. For this reason, they are less likely to overfit and are robust. Thus a nice side effect of being simple is better generalization.

Many researchers (e.g., in the heuristics-and-biases camp) often judge the quality of decisions by *coherence criteria* that derive from the laws of logic and probability. For example, preferences are supposed to be consistent and transitive. These include the standard assumptions on consumer preferences found in texts on discrete choice or microeconomics (e.g., see Chapter 3 of Ben-Akiva and Lerman [2] or Chapter 3 of Pindyck and Rubinfeld [53]). However, satisfying these normative characteristics does not guarantee effectiveness in the real world. Instead, *correspondence criteria* relate decision making strategies to performance in the external world. It turns out that fast and frugal heuristics (though sometimes viewed as irrational due to coherence criteria violations) are truly "rational" in an ecological sense when evaluated according to correspondence criteria.

Czerlinski et al [12] provide a good deal of evidence that many naturally occuring problems have a structure that can be exploited by fast and frugal heuristics. Importantly. there are cases where heuristics outperform more complex models.

When are simple heuristics likely to be applied. Payne et al [52] explore many factors that affect which decision strategies are used in different contexts. They

fall into two broad categories: task effects and context effects. The use of noncompensatory decision making can be influenced by task effects such as the number of alternatives, the number of attributes, whether or not the decision maker is under time pressure, whether the response mode is choice (choosing between alternatives) or judgment (assigning values to alternatives), and how the information is displayed (e.g., how many choices shown at one time). Decision making strategies can also be affected by context effects (properties of the alternatives, attributes, choice sets, etc.), e.g., similarity of alternatives, attribute ranges, correlation among attributes, and framing effects.

Because which decision making strategy is used seems contingent on (and adapted to) the structure of the particular task, Gigerenzer et al [23], Payne et al [52], and others have introduced the metaphor of an *adaptive toolbox*. Choosing which heuristic to apply from the toolbox can depend of the amount of information (e.g., using Take The Last instead of Take The Best if no cue validities are available). Deciding which tool to use can also be affected by the tradeoff between effort and accuracy.

There is substantial evidence that noncompensatory heuristics are used in situations like those described above (e.g., tasks with a large number of alternatives and attributes). Examples include: Bettman et al [3]; Bettman and Park [4]; Bröder [9]; Einhorn [14]; Einhorn and Hogath [15]; Gigerenzer and Goldstein [20]; Hauser [28]; Hauser and Wernerfelt [30]; Johnson and Meyer [34]; Luce, Payne and Bettman [44]; Martignon and Hoffrage [45], Montgomery and Svenson [49]; Payne [51]; Payne et al [52]; Roberts and Lattin [54]; Shugan [58]; and Urban and Hauser [68].

Consumer choice. Consumer choice is an area where the use of compensatory models has become standard practice. Conjoint analysis is a marketing science technique for analyzing how people choose between options that vary along multiple dimensions. It has been heavily used for over thirty years. Green et al [26] provide a thorough overview of the history of conjoint analysis, including data collection options and estimation methods. Conjoint analysis has been a vital part of many success stories in which accurately learning about consumer preferences was critical for improved product design and ultimate financial success. For example, Wind et

al [69] describe the design of the Courtyard by Marriott hotel chain—a project that was a finalist for the 1988 Franz Edelman Award from INFORMS.

However, given the substantial and growing evidence for heuristic decision making, it is important to address the mismatch between models and reality in marketing science practice. Managerially speaking, detecting the use of choice heuristics can affect advertising, product design, shelf display, etc. Due to the robustness of heuristics explained above, incorporating noncompensatory heuristics into conjoint analysis studies may also help increase the predictive ability for market simulators etc.

Inferring choice heuristics. In order to detect the use of choice heuristics, researchers have used verbal process tracing or specialized information processing environments (e.g., Mouselab or Eyegaze) to determine what process subjects used during a task. Payne et al [52] provide a review of how researchers have studied how consumers adapt or construct their decision processes during tasks.

Some software packages include steps in which respondents are asked to eliminate unacceptable levels (e.g., Srinivasan and Wyner's Casemap [64] and Johnson's Adaptive Conjoint Analysis [36]). However, because asking respondents outright to identify unacceptable levels (a form of screening rule) is sometimes problematic (Green et al [25], Klein [38]), some researchers have tried to infer the elimination process as part of the estimation (DeSarbo et al. [13], Gilbride and Allenby [24], Gensch [18], Gensch and Soofi [19], Jedidi and Kohli [32], Jedidi et al [33], Kim [37], Roberts and Lattin [54], and Swait [65]).

Bröder [9] uses statistical hypothesis tests to test for the Dawes equal weights compensatory model and a lexicographic noncompensatory model (where partworths have the form 1, $\frac{1}{2}$, $\frac{1}{4}$, etc.). Bröder's approach is conceptually similar to ours in that we are both interested in classifying subjects as compensatory or noncompensatory. However, in the empirical tests reported in [9], it appears that the approach was unable to classify a large portion of the respondents.

The approach closest to the proposed work is found in Kohli and Jedidi [39]. They analyze several lexicographic strategies and suggest a greedy heuristic for optimizing an otherwise hard integer programming problem. Our approach differs in several

ways, including data collection, algorithms, and focus, although it appears that both research teams hit on a similar development in parallel.

1.3 Contribution and outline

We attempt to alleviate the mismatch between theory and practice in conjoint analysis by providing a direct (and unobtrusive) way of estimating noncompensatory decision processes from data. The following outline highlights the main contributions of the thesis.

Chapter 2 We introduce several new lexicographic models for decision making. We also propose a constrained compensatory model that can aid in detecting (or ruling out) a truly compensatory process. We perform a simulation study to show that the constrained model has sufficient discriminatory ability.

Chapter 3 We propose several problems related to lexicographic (noncompensatory) inference and analyze their computational complexity. In particular, we show that some are easy due to possessing a greedoid language structure, while some are sufficiently hard that they admit no constant factor approximation scheme unless an unlikely condition holds.

Chapter 4 We construct exact greedy algorithms for the easy problems of Chapter 3. For the hard problems, we exploit some additional structure to formulate a dynamic programming recursion. While still having exponentially worst case runtime complexity, the dynamic programming algorithm is enhanced to perform very well in practice. Additionally, several local search heuristics are developed and analyzed. Numerical experiments explore the performance of the heuristics and the DP algorithm on the core lexicographic inference problem.

Chapter 5 We conduct an empirical study of SmartPhone preferences as a test of the effectiveness of the algorithms developed in Chapter 4, as well as exploring various behavioral questions. We find that a large portion of the respondents

behaved in a way consistent with lexicographic decision processes. In addition, the lexicographic strategies had better predictive ability on holdouts than two other benchmark compensatory models. We also analyze a dataset of computer preferences generously provided by another group of researchers and again find that the behavior of a significant portion of individuals can be explained by lexicographic models.

Chapter 6 Finally, we generalize lexicography and apply it to consideration set formation by allowing rules (logical expressions over features) for acceptance or rejection. We show that the problem of estimating rule sets given data is NP-hard. We develop an algorithm that can find the best rule sets of varying complexities using integer programming. We then apply the technique to the SmartPhone dataset and find that rule-based models for consideration predict as well or better than pure lexicographic or compensatory based approaches.

Using the techniques developed in this thesis, a researcher performing a study can perform indivual level estimation of lexicographic processes from observed data. Then, coupled with a standard compensatory analysis (possibly with constrained compensatory), the researcher can analyze to what extent noncompensatory processes were being used for the decision task and can segment the sample accordingly. This extra dimension of analysis can then aid in future product development, advertising, etc.

Chapter 2

Decision Making Models

In this chapter, we present several noncompensatory decision processes in the lexicographic family. We propose a constrained compensatory model to be used as an aid for gauging whether or not a process is truly compensatory. Finally, we test the ability of the constrained formulation to help rule out compensatory models.

2.1 Notation

Following Tversky [67], we use the term *aspect* to refer to a binary feature, e.g., "big" or "has-optical-zoom". We will typically denote aspects using lower case letters such as a, b, c, or subscripted as, e.g., a_1, a_2, a_3. The set of all aspects will sometimes be denoted by A. Note that k-level features, e.g., low, medium, and high price, can be coded as k individual aspects: "low-price", "medium-price", and "high-price".

We use the term *profile* to refer to a set of aspects, representing a particular product (or other) configuration. For example, a digital camera profile might be

$$\{\text{low-price, small, has-optical-zoom, 1-megapixel}\}.$$

Profiles will typically be denoted as P_i, with the set of all profiles being P. An aspect a is said to *differentiate* two profiles P_i and P_j if and only if a is contained in exactly one of the profiles.

Given a set of profiles, preferences can be elicited from a subject in several ways. Profiles can be rank ordered. The subject can select a subset for consideration and subsequently rank order only those selected for serious consideration. In a choice-based conjoint analysis setting, the subject is presented with a series of small sets of profiles and asked which one is most preferred in each. In a metric setting, the subject is asked to provide a rating for each profile, say from 1 to 100.

In all cases, a set of paired comparisons can be generated,

$$\Omega = \{(P_i, P_j) : P_i \succ P_j\}.$$

Here, $P_i \succ P_j$ means that P_i is preferred to P_j. We use the notation $P_i \approx P_j$ to indicate that neither $P_i \succ P_j$ nor $P_j \succ P_i$. We will also occasionally add subscripts to the preference and indifference symbols to make it clear where the preferences came from (if necessary), e.g., $P_i \succ_\Omega P_j$. If the preference relation on profiles is reflexive, antisymmetric, and transitive, it can be viewed as a *partial order* on the profiles, which we will denote by X. In the case of rank order data, we actually have a total (linear) order over the profiles.

An *aspect order* is an ordered set of aspects and will typically be denoted by α. The following is some notation for identifying key characteristics and features of an aspect order.

$I_\alpha(a)$: position (or index) of aspect a in α

$\alpha(i)$: aspect in position i

$\alpha^{(k)}$: left subword of α of length k

$= (\alpha(1), \alpha(2), \ldots, \alpha(k))$, i.e., the left subset of α of length k

The following definitions relate to special sets of aspects or refer to special aspects:

$A^{\succ}(P_i, P_j)$: set of aspects that are in P_i but not in P_j

$= \{a \in A : a \in P_i, a \notin P_j\}$

$A^{\prec}(P_i, P_j)$: set of aspects that are not in P_i but are in P_j

$= \{a \in A : a \notin P_i, a \in P_j\}$

$A^0(P_i, P_j)$: set of aspects that do not differentate P_i and P_j

$$= A \setminus (A^{\succ}(P_i, P_j) \cup A^{\prec}(P_i, P_j))$$

$f_\alpha(P_i, P_j)$: first (leftmost) aspect in α that differentiates P_i and P_j

$$= \arg\min_{a \in A^{\succ}(P_i,P_j) \cup A^{\prec}(P_i,P_j)} I_\alpha(a)$$

Given an aspect order, the *lexicographic preference relation* over profiles is given by $\succ_\alpha$. In this relation,

$$P_i \succ_\alpha P_j \Leftrightarrow f_\alpha(P_i, P_j) \in A^{\succ}(P_i, P_j)$$

That is, P_i is lexicographically preferred to P_j if and only if the first aspect that differentiates P_i and P_j is contained in P_i.

Note that using a lexicographic decision rule to make decisions between profiles is fast because there are no computations involved (besides noting the presence or absence of aspects). It is also frugal because once a differentiating aspect has been found, no subsequent aspects from the aspect order need to be considered. The process is noncompensatory because the final decision (i.e., which profile is preferred) depends solely on the most important differentiating aspect—the presence of less important aspects can never compensate for the absence of a more important aspect.

The following sets and functions relating X and $\succ_\alpha$ will be helpful in later analysis.

X_α^+ : set of all pairs in X that are differentiated correctly by α

$$= \{(P_i, P_j) \in X : P_i \succ_\alpha P_j\}$$
$$= \{(P_i, P_j) \in X : f_\alpha(P_i, P_j) \in A^{\succ}(P_i, P_j)\}$$

X_α^- : set of all pairs in X that are differentiated incorrectly by α

$$= \{(P_i, P_j) \in X : P_i \prec_\alpha P_j\}$$
$$= \{(P_i, P_j) \in X : f_\alpha(P_i, P_j) \in A^{\prec}(P_i, P_j)\}$$

X_α^0 : set of all pairs in X that are not differentiated by α

$$= X \setminus (X_\alpha^+ \cup X_\alpha^-)$$

$E_X(\alpha)$: number of errors/violations caused by α

$$= |\{(P_i, P_j) \in X : P_i \prec_\alpha P_j\}|$$

$M_X^-(\alpha)$: number of (new) errors/violations caused by the last aspect in α

$$= |\{(P_i, P_j) \in X : \alpha^{(k-1)} \cap A^0(P_i, P_j) = \emptyset \text{ and } \alpha(k) \in A^\prec(P_i, P_j)\}|$$

$M_X^+(\alpha)$: number of (new) correct differentiations caused by the last aspect in α

$$= |\{(P_i, P_j) \in X : \alpha^{(k-1)} \cap A^0(P_i, P_j) = \emptyset \text{ and } \alpha(k) \in A^\succ(P_i, P_j)\}|$$

Finally, we define the term *lexico-consistent* with respect to both aspect orders and data.

Definition 1 *We say that an aspect order α is* lexico-consistent *with a partial order on profiles X if $E_X(\alpha) = 0$.*

Definition 2 *We say that a partial order on profiles X is* lexico-consistent *if there exists an aspect order α such that $E_X(\alpha) = 0$.*

2.2 Lexicographic Models

In the previous section, it was implicitly assumed that the presence of an aspect was considered "good". Following Tversky's nomenclature in [67], we call a decision process based on the lexicographic preference relation $\succ_\alpha$ *acceptance-by-aspects* (ABA). For example, suppose we have a product category with 5 features—one four-level feature, one three-level feature, and three two level (binary) features.

$$a_1, a_2, a_3, a_4, b_1, b_2, b_3, c, d, e$$

Then a possible ABA strategy is

$$\alpha_{ABA} = (a_1, a_4, b_3, d, e, c, a_2, b_2, b_1, a_3)$$

However, it is not always clear whether inclusion or exclusion of an aspect is

preferred. Thus it will sometimes be necessary to refer to the *orientation* of an aspect. If the opposite (or absence) of aspect a is preferred, the subject will be said to prefer $\neg a$ (or reject a).

ABA is intimately connected to the deteriministic version of Tversky's elimination-by-aspects (EBA). In EBA, all profiles that contain the most important aspect $\alpha(1)$ are eliminated. Next, all profiles containing the secondmost important aspect $\alpha(2)$ are eliminated, and so on until only one profile remains. This is equivalent to an ABA process where α contains negated versions of each aspect (though in the same positions):

$$\begin{aligned} & \alpha_{EBA} = (a_1, a_4, b_3, d, e, c, a_2, b_2, b_1, a_3) \\ \Leftrightarrow\ & \alpha_{ABA} = (\neg a_1, \neg a_4, \neg b_3, \neg d, \neg e, \neg c, \neg a_2, \neg b_2, \neg b_1, \neg a_3) \end{aligned}$$

Alternatively, EBA can be viewed as ABA with aspects recoded to their opposite orientations (e.g., change aspect "big" to "small").

We generalize ABA and EBA to allow the mixing of acceptance and rejection rules, which we refer to as lexicographic-by-aspects (LBA). In this case, any aspect can occur in either the accept or reject orientation, e.g.,

$$\alpha_{LBA} = (a_1, \neg a_4, b_3, \neg d, e, c, a_2, \neg b_2, b_1, a_3)$$

We say an aspect order is implementing a lexicographic-by-feature (LBF) decision process if the aspects corresponding to each multi-level feature are grouped together. This is equivalent to first ranking the features according to importance and then ranking the levels within each feature, e.g.,

$$\alpha_{LBF} = (b_3, b_1, b_2, c, e, a_1, a_4, a_2, a_3, d)$$

Finally, notice that when all features are binary, ABA, EBA, LBA, and LBF are all equivalent (assuming we always allow recoding of binary features since their orientation is arbitrary). On the other hand, when there are one or more multi-level

features, the models are not strictly equivalent. For example, the EBA strategy

$$\alpha_{EBA} = (\$high, green, small, \$medium, red, blue, \$low)$$

has no exact ABA counterpart. If we allowed unions of aspects (see Section 3.2.4), an equivalent ABA model would be

$$\alpha_{ABA} = (\$low \text{ or } \$medium, red \text{ or } blue, big, \$low, blue, green, \$high)$$

2.3 Compensatory and Constrained Compensatory Models

In a linear compensatory model (also known as Franklin's rule or a weighted additive model), there are partworths or weights w_i associated with each aspect a_i, with

$$P_i \succ P_j \Leftrightarrow \sum_{a \in P_i} w_i > \sum_{a \in P_j} w_i.$$

As Martignon and Hoffrage [45] show, compensatory models include lexicographic (noncompensatory) models as a special case. For a given aspect order α, any set of weights satisfying

$$w_{\alpha(i)} > \sum_{j=i+1}^{n} w_{\alpha(j)}, \forall i \tag{2.1}$$

will result in the same preference relation as $\succ_\alpha$. For example, this property is satisfied by setting

$$w_{\alpha(i)} = 2^{1-i}.$$

In order to determine whether or not these extreme lexicographic weights are necessary to fit a given respondent or not, we would like the ability to exclude partworth

vectors that are consistent with (2.1). A set of constraints such as

$$\begin{aligned} w_1 &< \textstyle\sum_{i \neq 1} w_i \\ w_2 &< \textstyle\sum_{i \neq 2} w_i \\ &\vdots \\ w_n &< \textstyle\sum_{i \neq n} w_i \end{aligned}$$

would prevent partworth vectors from satisfying (2.1). However, even though a single aspect would be prevented from dominating the rest, the rest of the aspects could have a perfectly lexicographic substructure. It is not clear how to prevent lexicographic substructures with a polynomial number of linear constraints. Thus we propose an alternative approach.

The form of our constraints is motivated by behavioral researchers who have sought to identify whether compensatory or noncompensatory models fit or predict observed choices better. For example, Bröder [9] requires that $w_i = w_j$ for all i, j. We generalize Bröder's constraint by defining a set of partworths as *q-compensatory* if $w_i \leq q w_j$ for all $i \neq j$. With this definition, we can example a continuum between Dawes' model as tested by Bröder ($q = 1$) and the unrestricted additive benchmark ($q = \infty$) that nests lexicographic models.

It is interesting to note that Dawes' Rule ($w_i = 1$ for all i) can be considered to be both the most compensatory weighted additive model *and* a simplifying heuristic. Because of the special nature of the weights, simple counting can be used for determining the overall utility of a profile versus computing a weighted sum in the general compensatory case (which requires multiplication and addition operations).

Monte Carlo Simulation. The q-compensatory constraints can be incorporated into most existing compensatory partworth estimation techniques. For example, LINMAP (Srinivasan and Shocker [63] and Srinivasan [62]) uses a linear program to find partworths that optimize an objective function related to the set of paired comparisons. Analytic center methods (Toubia et al [66]) find the analytic center of a particular polyhedron (which is defined by linear constraints). Hierarchical Bayes (HB) techniques (e.g., Rossi and Allenby [56]) uses a hierarchy of probability distri-

butions so that population data can inform and improve individual level estimation. The sampling (random draws) over partworth distributions can be restricted to the q-compensatory constrained region.

For simplicity of exposition, we report Monte Carlo results for LINMAP and for rank-order data only. We obtain qualitatively similar results for consider-then-rank synthetic data. For these tests, we use the 32×16 SmartPhone experimental design that will be described in Chapter 5.

For our generating model, we modify a functional form proposed by Einhorn [14]. We first define a set of generating weights, $\omega_n = 2^{1-n}$ for $n = 1$ to N. We then select each synthetic respondent c's true partworths as follows: $w_{nc} = (\omega_n)^m = 2^{m(1-n)}$ for the nth smallest partworth. Following Einhorn, $m = 0$ implies Dawes' model and $m = 1$ implies a minimally lexicographic model. (By minimally lexicographic, we mean that the model may not be lexicographic in the presence of measurement error.) Setting $0 < m < 1$ generates a q-compensatory model. By setting $m = 0$, $1/15$, $2/15$, $4/15$, $8/15$, and $16/15$ we generate a range of models that are successively less compensatory. (For 16 aspects, the smallest partworth is 2^{-15}. Setting the largest m to $16/15$ makes the last model less sensitive to measurement error.) We then generate $1,000$ synthetic respondents for each m as follows where u_{jc} is respondent c's true utility for profile j.

1. For each m, generate w_c, normalize so w_{nc}'s sum to 1.0.

2. For each c, add error to each true profile utility: $\tilde{u}_{jc} = u_{jc} + \tilde{\epsilon}_j$, where $\tilde{\epsilon}_j \backsim N(0, e)$ and $e = 0.2, 0.4$.

3. Given $\{u_{kc}\}$, generate a rank order of 32 cards for respondent c. Repeat for all m.

For each respondent, we use estimate an LBA aspect order (using algorithms developed in Chapter 4) and use LINMAP(q) to estimate q-compensatory partworths. Both aspect orders and estimated partworths imply a rank order of the 32 profiles. The comparison statistic is the percent of ordered pairs of profiles predicted from

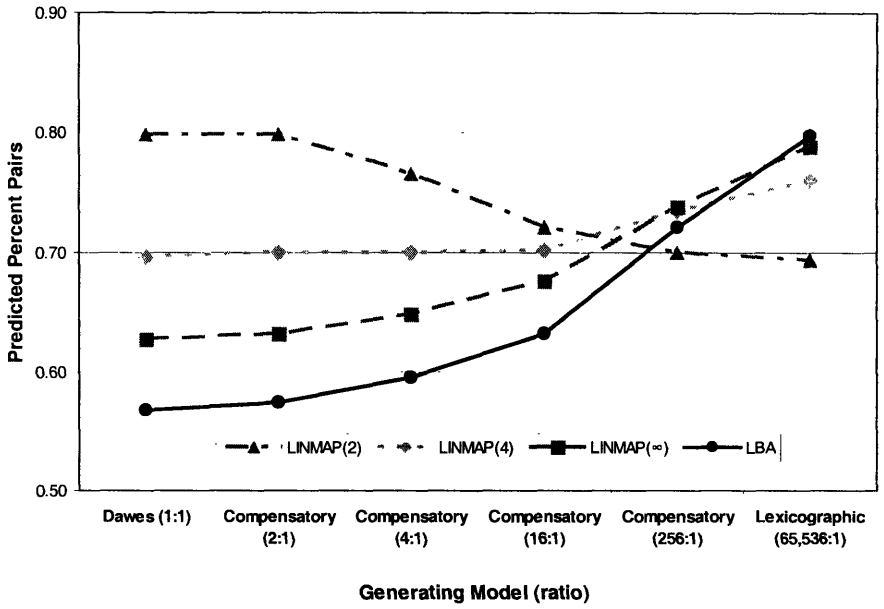

Figure 2-1: Results of the Monte Carlo Experiments

the estimated model that are consistent with the true model. The results are shown in Figure 2-1. For ease of interpretation and comparison with the q-compensatory constraints, we label the horizontal axis with the ratio of the largest to the smallest partworth. For example, $m = 2/15$ implies a ratio of $4:1$.

Compare first the highly constrained compensatory model, LINMAP(2), to LBA. As expected, the compensatory model predicts better than LBA when respondents are truly compensatory and LBA predicts better than LINMAP(2) when respondents are truly lexicographic. Furthermore, there is a smooth movement from LINMAP(2) to LINMAP(∞) as q increases. This is also true for $q = 1$, 8 and 16 (not shown for simplicity). For this particular simulation with homogeneous respondents, constraints help significantly for low m. The unconstrained compensatory model, LINMAP(∞) may overfit the data for low m. We expect this to be mitigated with heterogeneous respondents as will be seen in the SmartPhone and computer empirical studies in Chapter 5. Finally, we see that $q = 4$ is a reasonable discriminator vs. LBA because the two curves cross for m in a moderate range.

Chapter 3

Complexity Analysis

In this chapter, we introduce several problems related to noncompensatory inference. We analyze the computational complexity of each problem, showing whether they belong to the class of problems that can be solved with polynomial-time algorithms or belong to more difficult classes. Several problems are shown to be easy by proving that they have a greedoid language structure. Other problems are shown to be hard and, furthermore, hard to approximate.

3.1 NP-Completeness and Approximation

Computational Complexity. For proving properties about the hardness of various problems, we rely on the theory of computational complexity. In the 1970s, researchers began developing the theory of NP-completeness and studying other properties of complexity classes, i.e., classes of problems with the same level of difficulty. Garey and Johnson [17] is the standard text on NP-completeness.

Problems in class P can be solved with polynomial-time algorithms. Problems in class NP have the property that a solution can be verified in polynomial time (e.g., checking whether a given traveling salesman tour has length less than k). Finally, a problem is in the class NP-complete if it is in NP and also has the property that any other problem in NP can be transformed to it with a polynomial-time transformation. Examples of problems that are NP-complete include VERTEX COVER,

SATISFIABILITY, and the TRAVELING SALESMAN PROBLEM. The class NP-complete contains essentially the "hardest" problems in NP (since any algorithm for a NP-complete problem can be applied (after transformation) to any other problem in NP). Specifically, if a polynomial-time algorithm were discovered for an NP-complete problem, it would imply that P = NP and all problems in NP would be polynomially (efficiently) solvable.

The initial framework for studying computational complexity was based in logic, and the problems were all cast as decision problems (i.e., problems that asked a yes/no question). Thus, instead of asking what the smallest (minimum size) vertex cover is, it is asked if there exists a vertex cover of size less than or equal to k.

Subsequent work has extended and applied complexity analysis more directly to optimization problems (see Ausiello et al [1]). Additionally, even though the class NP-complete contains many hundreds of equally hard problems in the decision sense, not all problems are equally hard when it comes to approximability. For example, certain problems admit approximation schemes that guarantee a solution within a factor r of optimality, while others do not. The problems that do have constant-factor approximation algorithms belong to the class APX. Thus, even after showing that an optimization problem is NP-hard (i.e., that all problems in NP can be transformed to it in polynomial time, though the problem itself is not necessarily in NP), it is often useful to determine how difficult it is to approximate it.

3.2 Easy Problems

3.2.1 Greedoid languages

Greedoids are mathematical objects initially developed by Korte and Lovasz [40] to study conditions under which a greedy algorithm can solve optimization problems. They have proven useful in sequencing and allocation problems (e.g., Niño-Mora [50]). Björner and Ziegler [5] and Korte et al [41] are excellent references that provide numerous examples of greedoids. We believe that this is the first application of

greedoids to marketing science.

Greedoids are a class of set systems that possess certain properties. (Any matroid, a perhaps more widely known object, is also a greedoid.) It can be shown that greedoids have an equivalent language representation. The language form given in Definition 3 is more appropriate for our application.

Definition 3 *A greedoid language* (E, L) *is an alphabet* E *and a language* L *such that*

(G1) *If* $\alpha \in L$ *and* $\alpha = \beta\gamma$, *then* $\beta \in L$.

(G2) *If* $\alpha, \beta \in L$ *and* $|\alpha| > |\beta|$, *then there exists an* $x \in \alpha$ *such that* $\beta x \in L$.

Property (G1) means that if a word is in the language, then any left subword must also be in the language. Property (G2) states that if two words are in the language, then there exists some letter from the larger word that can be right appended to the shorter word to make a new word in the language.

Greedoids (and greedoid languages) are important since for a certain class of *compatible* objective functions (see Definition 4), the greedy algorithm is guaranteed to return optimal solutions.

Definition 4 *(from Boyd [7]) An objective function* W *is* compatible *with a language* L *if the following conditions hold: If* $\alpha x \in L$ *and* $W(\alpha x) \geq W(\alpha y)$ *for all* y *such that* $\alpha y \in L$ *(i.e.,* x *is the best choice at* α*) then*

(a) $\alpha\beta x\gamma \in L, \alpha\beta z\gamma \in L \Longrightarrow W(\alpha\beta x\gamma) \geq W(\alpha\beta z\gamma)$
(x is best at every later stage)

(b) $\alpha x\beta z\gamma \in L, \alpha z\beta x\gamma \in L \Longrightarrow W(\alpha x\beta z\gamma) \geq W(\alpha z\beta x\gamma)$
(x before z is always better than z before x)

For several problems in this chapter, we will be interested in finding the longest word in the greedoid language. Lemma 1 shows that this objective function is compatible.

Lemma 1 *The objective function $W(\alpha) = |\alpha|$ is compatible with any language L.*

Proof. Property (a). Suppose $\alpha\beta x\gamma \in L$, $\alpha\beta z\gamma \in L$. Then

$$W(\alpha\beta x\gamma) = |\alpha\beta x\gamma| = |\alpha\beta z\gamma| = W(\alpha\beta z\gamma).$$

Property(b). Suppose $\alpha x\beta z\gamma \in L$, $\alpha z\beta x\gamma \in L$. Then

$$W(\alpha x\beta z\gamma) = |\alpha x\beta z\gamma| = |\alpha z\beta x\gamma| = W(\alpha z\beta x\gamma).$$

Thus W is a compatible objective function. □

3.2.2 Is there an aspect order that is lexico-consistent with the data?

Here we define the problem LEX CONSISTENCY in the style of Garey and Johnson [17] by giving the objects that make up an instance along with the decision question. For the optimization problems that appear later in this section, we follow the style of Ausiello et al [1] by giving the form of the instance, the form of a feasible solution, and the measure (over feasible solutions) to be optimized.

LEX CONSISTENCY
INSTANCE : Set of aspects A, set of profiles P, partial order on profiles X
QUESTION : Is there an aspect order α such that $E_X(\alpha) = 0$?

The first problem we consider is one of the two core noncompensatory inference problems. Given data, e.g., a set of pairs of profiles Ω or a partial order on the profiles X, we are interested in determining if there exists an aspect order that is lexico-consistent with the data, i.e., that induces no errors with respect to the data. We first show that this problem has a special greedoid language structure, and then show that the problem is in complexity class P.

Theorem 1 *Let X be a partial order on the profiles P, and let G be the collection of all aspect orders that are consistent with X. Then G is a greedoid language.*

Proof. Property (G1). Suppose $\alpha = \beta\gamma \in G$. Consider any $(P_i, P_j) \in X$ and let $x = f_\alpha(P_i, P_j)$ be the first aspect in α (if any) that differentiates P_i and P_j. If $x \in \beta$, then $f_\beta(P_i, P_j) = x$ and $P_i \succ_\beta P_j$. If $x \notin \beta$, then $f_\beta(P_i, P_j) = \emptyset$ (by definition of x) and β is consistent with $P_i \succ_X P_j$.

Property (G2). Suppose $\alpha, \beta \in G$ with $|\alpha| > |\beta|$. Let $x \in \alpha$ be the first aspect from α that is not also in β, i.e.,

$$x = \arg\min_{a \in \alpha \setminus \beta} I_\alpha(a),$$

and consider the new word βx. For any $(P_i, P_j) \in X$, there are two cases to consider. (1) If either $P_i \succ_\beta P_j$ or both $P_i \approx_\beta P_j$ and $x \notin A^-(P_i, P_j)$, then βx is consistent with $P_i \succ_X P_j$. (2) Suppose that $P_i \approx_\beta P_j$, and, for the sake of contradiction, that $x \in A^-(P_i, P_j)$. This implies that $f_\alpha(P_i, P_j) \neq x$ (since $\alpha \in G$), and there exists an aspect $x' \in \alpha$ such that $f_\alpha(P_i, P_j) = x'$ and $I_\alpha(x') < I_\alpha(x)$. By the definition of x, $x' \in \beta$, which contradicts $f_{\beta x}(P_i, P_j) = x$. Thus $x \notin A^-(P_i, P_j)$ and βx is consistent with $P_i \succ_X P_j$. □

Corollary 1 *LEX CONSISTENCY is in P.*

Proof. Since G is a greedoid language and $W(\alpha) = |\alpha|$ is compatible, the greedy algorithm (which has polynomial running time) can be used to find the longest word in G. The partial order X is lexico-consistent if and only if the maximum word length returned by the greedy algorithm is n. Thus, we can determine if X is lexico-consistent in polynomial time. □

3.2.3 Is there an aspect order such that each aspect introduces at most k new errors?

The previous section considered whether there existed an aspect order that was perfectly lexico-consistent with the data. Here we relax the objective. Instead of desiring

an aspect order where each aspect introduces zero new errors, we consider aspect orders where each aspect is permitted to cause up to k errors.

BOUNDED ERRORS PER ASPECT	
INSTANCE :	Set of aspects A, set of profiles P, partial order on profiles X, scalar k
QUESTION :	Is there an aspect order α such that $M_X(\alpha(i)) \leq k$ for all i?

We first show that the problem has a greedoid language structure, and then show that the problem is in class P.

Theorem 2 *Let X be a partial order on the profiles, and let G be the collection of all aspect orders such that each aspect introduces at most k (new) errors with respect to X. Then G is a greedoid language.*

Proof. Property (G1). Suppose $\alpha = \beta\gamma \in G$. By the definition of G, we have $M_X(\alpha^{(i)}) \leq k$, for $i = 1, \ldots, n$. Since $\beta = \alpha^{(j)}$ for some $j < n$, we have $M_X(\beta^{(i)}) = M_X(\alpha^{(i)})$, for $i = 1, \ldots, j$, showing that $\beta \in G$.

Property (G2). Suppose $\alpha, \beta \in G$ and $|\alpha| > |\beta|$. Let $x \in \alpha$ be the first aspect from α that is not also in β, i.e.,

$$x = \arg\min_{a \in \alpha \setminus \beta} I_\alpha(a),$$

and consider the new word βx. We will show that $M_X(\beta x) \leq k$. Let $\alpha' = \alpha^{(I_\alpha(x))}$, i.e., the left subword of α up to and including x. For all $(P_i, P_j) \in X$ such that either $P_i \succ_\beta P_j$ or both $P_i \approx_\beta P_j$ and $x \notin A^-(P_i, P_j)$, the relation $P_i \succ_X P_j$ is not violated and so does not contribute to $M_X(\beta x)$. Suppose instead that $P_i \succ_X P_j$ while $P_i \approx_\beta P_j$ and $x \in A^-(P_i, P_j)$. By the definition of x, we have

$$f_{\beta x}(P_i, P_j) = f_{\alpha'}(P_i, P_j) = x.$$

(If there was an $x' \in \alpha \setminus \beta$ with smaller index than x that also differentiated P_i and P_j, it would have been chosen instead of x.) This means that $P_j \succ_{\alpha'} P_j$ since $x \in A^-(P_i, P_j)$. We have shown that if βx violates $P_i \succ_X P_j$, then α' also violates $P_i \succ_X P_j$. Thus, $M_X(\beta x) \leq M_X(\alpha') \leq k$. □

Corollary 2 *BOUNDED ERRORS PER ASPECT is in P.*

Proof. Since G is a greedoid language and $W(\alpha) = |\alpha|$ is compatible, the greedy algorithm (which has polynomial running time) can be used to find the longest word in G. There exists an aspect order such that each aspect introduces at most k new errors if and only if the maximum word length returned by the greedy algorithm is n, which can be determined in polynomial time. □

3.2.4 Adding unions and intersections of aspects

In regression and other methods that involve modeling with independent (or predictor) variables, it is often necessary and/or advantageous to include interactions between variables or other nonlinear derived terms. Here we consider adding derived aspects that are formed from unions and intersections over all pairs of aspects.

For example, if the original set of aspects included { small, medium, large } and { red, blue, green }, then possible derived aspects would include small-and-red, medium-or-green, and blue-or-green. We show that exanding the set of aspects in this way does not affect the greedoid structure with respect to lexico-consistency.

Theorem 3 *Let A be a set of aspects, P be a set of profiles, and X be a partial order on the profiles P. Let $A' = A \cup \{a_{i \cup j}\} \cup \{a_{i \cap j}\}$, i.e., the original aspects plus all possible unions and intersections of two aspects, and let G be the collection of all aspect orders over A' that are consistent with X. Then G is a greedoid language.*

Proof. This follows immediately from Theorem 1. Replacing aspect set A with A' does not change the structure of the problem. □

3.3 Hard Problems

In this section, we consider problems that are not in class P (unless $P = NP$). In many of the proofs, we will reduce the problems to MIN SET COVER, a canonical problem in approximation.

MIN SET COVER
INSTANCE : Collection C of subsets of a finite set S.
SOLUTION : A set cover for S, i.e., a subset $C' \subseteq C$ such that every element in S belongs to at least on member of C'
MEASURE : Cardinality of the set cover, i.e., $\|C'\|$.

3.3.1 Minimum number of errors

Suppose we know that there is no aspect order lexico-consistent with the data. In that case, we might still be interested in the aspect order that induces the least number of errors with respect to the data, i.e., that fits it best. We refer to this problem as MIN LEX ERRORS.

MIN LEX ERRORS
INSTANCE : Set of aspects A, a set of profiles P, and a set of pairs $X \subseteq P \times P$.
SOLUTION : Aspect order α such that $\|\alpha\| = \|A\|$.
MEASURE : $E_X(\alpha)$

Unlike LEX CONSISTENCY, MIN LEX ERRORS is not in P (unless $P = NP$). Schmitt and Martignon [57] show that the decision version of this problem is NP-complete with a reduction from VERTEX COVER. Here we strengthen their result by showing that not only is MIN LEX ERRORS NP-hard, but it is AP-reducible to MIN SET COVER. The following result is known for MIN SET COVER.

Theorem 4 (Feige [16]) *MIN SET COVER is not approximable within* $(1-\delta)\ln n$ *for any* $\delta > 0$ *unless* $NP \subset DTIME(n^{O(\log\log n)})$, *where* $DTIME(t)$ *is the class of problems for which there is a deterministic algorithm running in time* $O(t)$.

By reducing a problem from MIN SET COVER with an approximation preserving (AP) reduction, the problem is shown to be at least as hard to approximate as MIN SET COVER. From Feige's result, that means there can be no constant factor approximation algorithm unless $NP \subset DTIME(n^{O(\log\log n)})$ (which is unlikely).

Theorem 5 *MIN SET COVER is AP-reducible to MIN LEX ERRORS.*

Proof. Given a collection C of subsets of set S in a MIN SET COVER instance, we construct a MIN LEX ERRORS instance as follows. Let $C_1, C_2, \ldots, C_n$ be the elements of C and let $x_1, x_2, \ldots, x_m$ be the elements of S. Associate an aspect a_i with each element C_i of C. Introduce an additional aspect q. Associate a profile P_i with each element x_i of S such that

$$P_i = \{a_j : x_i \in C_j\}$$

Introduce new profiles V and $W_1, W_2, \ldots, W_n$ such that

$$V = \{q\}$$
$$W_i = \{a_i\}.$$

Finally, let

$$X = \{(P_i, V) : i = 1, \ldots, m\} \cup \{(V, W_i) : i = 1, \ldots, n\}.$$

Note that the (P_i, V) pairs in X will all be correctly differentiated by α if x_i is contained in some C_j such that a_j comes before q in α (i.e., if α corresponds to an actual set cover). The (V, W_i) pairs in X encourage q to appear as left as possible in α. Figure 3.3.1 shows an example transformation. Notice that the transformation can be accomplished in polynomial time.

	C_1	C_2	C_3	C_4
x_1	1	0	1	1
x_2	0	1	1	0
x_3	0	1	0	1
x_4	1	0	0	0
x_5	0	0	1	0
x_6	1	0	0	1
x_7	1	1	0	0

$\Longrightarrow$

	a_1	a_2	a_3	a_4	q
P_1	1	0	1	1	0
P_2	0	1	1	0	0
P_3	0	1	0	1	0
P_4	1	0	0	0	0
P_5	0	0	1	0	0
P_6	1	0	0	1	0
P_7	1	1	0	0	0
V	0	0	0	0	1
W_1	1	0	0	0	0
W_2	0	1	0	0	0
W_3	0	0	1	0	0
W_4	0	0	0	1	0

Figure 3-1: Transformation from MIN SET COVER to MIN LEX ERRORS

The errors/violations for this MIN LEX ERRORS instance are of two types:

(V1) Violations of $P_i \succ V$

(V2) Violations of $V \succ W_i$

Let the number of violations of type V1 and V2 for the approximate solution be e_1 and e_2. Next we show that the best MIN LEX ERRORS can achieve is $e_1 + e_2 = k^*$, where k^* is the size of a minimum set cover in the original MIN SET COVER instance.

Let $\alpha = \beta q \gamma$ be an aspect order. If β corresponds to a set cover in the original MIN SET COVER instance, then $e_2 = |\beta|$ and $e_1 = 0$. On the other hand, if β does not correspond to a set cover in the original MIN SET COVER instance, then $e_2 = |\beta|$ and e_1 is the number of elements from S not covered by the C_j corresponding to β. Note that $e_1 \geq |\delta|$, where δ is the smallest subset of γ such that $\beta\delta$ corresponds to a set cover in the original MIN SET COVER instance. Combining these two cases, we see that $e_1 + e_2$ is always at least as large as the size of the smallest set cover containing β. Thus the best objective value MIN LEX ERRORS can achieve is to have β correspond to a minimum set cover in the original MIN SET COVER instance.

Now suppose we have an r-approximation algorithm for MIN LEX ERRORS. This guarantees that

$$\frac{e_1 + e_2}{k^*} \leq r,$$

where k^* is the size of the minimum set cover of the original problem. To transform an r-approximate solution to MIN LEX ERRORS into a set cover for the original MIN SET COVER instance, consider the following. If the C_j corresponding to the a_j in β do not already form a set cover for the original MIN SET COVER instance, then at most e_1 additional elements from C need to be added to the existing group of e_2 elements to form a set cover (since e_1 corresponds to the number of elements of S not yet covered). The size of the constructed set cover is at most $e_2 + e_1$, guaranteeing a performance ratio

$$\frac{e_2 + e_1}{k^*} \leq r.$$

Therefore, MIN SET COVER is AP-reducible to MIN LEX ERRORS. □

3.3.2 Minimum weighted number of errors

It might be the case that for a given set of pairs of profiles Ω, some comparisons are more important than others. For example, perhaps the comparisons from the earlier part of a conjoint survey are deemed more likely to be accurate than later comparisons when a subject might be more tired.

MIN WEIGHTED ERRORS
INSTANCE : Set of aspects A, set of profiles P, set of pairs $X \subseteq P \times P$ along with weights w_{ij} for all $(P_i, P_j) \in X$.
SOLUTION : Aspect order α such that $\lvert\alpha\rvert = \lvert A\rvert$.
MEASURE : $E_{X,w}(\alpha)$

Theorem 6 *MIN WEIGHTED ERRORS is NP-hard.*

Proof. This result follows directly from Theorem 5 since by setting all weights w_{ij} equal to 1, MIN WEIGHTED ERRORS is equivalent to MIN LEX ERRORS. □

3.3.3 Minimum position of an aspect given lexico-consistent

Given that a set of data is lexico-consistent, we might be interested in the leftmost position that a particular aspect can occur in among all lexico-consistent aspect orders. Note that how far left an aspect occurs relates to the aspect's importance. We call this problem MIN ASPECT POSITION.

MIN ASPECT POSITION	
INSTANCE	: Set of aspects A, aspect $q \in A$, a set of profiles P, and a set of pairs $X \subseteq P \times P$ that is lexico-consistent.
SOLUTION	: Aspect order α with $E_X(\alpha) = 0$.
MEASURE	: $I_\alpha(q)$

Theorem 7 *MIN SET COVER is AP-reducible to MIN ASPECT POSITION.*

Proof. The reduction from MIN SET COVER is nearly the same as in the proof for MIN LEX ERRORS, except there is no need for profiles W_i. Suppose we have an r-approximate algorithm for MIN ASPECT POSITION, i.e.,

$$\frac{I_\alpha(q)}{k^* + 1} \leq r,$$

where k^* is the size of the minimum set cover of the original problem. Note that all feasible solutions to MIN ASPECT POSITION must be of the form $\alpha = \beta q \gamma$ where β corresponds to a set cover in the original MIN SET COVER instance (otherwise some $P_i \succ V$ would be violated). Thus, the best objective value MIN ASPECT POSITION can achieve is $k^* + 1$, by letting β be a minimum set cover in the original MIN SET COVER instance.

The C_j corresponding to the a_j in β (from the r-approximate solution to MIN ASPECT POSITION) form a set cover C' for the original MIN SET COVER since all pairs (P_i, V) are correctly differentiated by α. With respect to MIN SET COVER,

the performance ratio is

$$\frac{I_\alpha(q)-1}{k^*}.$$

Calculating the difference between the performance ratios of MIN SET COVER and MIN ASPECT POSITION,

$$\frac{I_\alpha(q)-1}{k^*}-\frac{I_\alpha(q)}{k^*+1}=\frac{I_\alpha(q)-(k^*+1)}{k^*(k^*+1)}=\frac{I_\alpha(q)}{k^*(k^*+1)}-\frac{1}{k^*}=\left(\frac{1}{k^*}\right)\left(\frac{I_\alpha(q)}{k^*+1}-1\right),$$

we see that

$$\frac{I_\alpha(q)-1}{k^*}=\frac{I_\alpha(q)}{k^*+1}+\left(\frac{1}{k^*}\right)\left(\frac{I_\alpha(q)}{k^*+1}-1\right)\leq r+(r-1),$$

since $k^* \geq 1$ and $\frac{I_\alpha(q)}{k^*+1} \leq r$. Therefore, MIN SET COVER is AP-reducible to MIN ASPECT POSITION. □

3.3.4 Minimum number of aspects needed to explain lexico-consistent partial order on profiles

Suppose we have a set of data that is lexico-consistent. For concreteness, suppose our data consists of a set of pairs Ω. Even though a full aspect order differentiates all possible pairs of profiles, it might be the case that a partial aspect order, i.e., with $|\alpha| < |A|$, can differentiate all pairs in Ω correctly. We call the problem of finding the shortest such aspect order (that is still lexico-consistent with the data) MIN ASPECTS TO EXPLAIN.

MIN ASPECTS TO EXPLAIN
INSTANCE : Set of aspects A, a set of profiles P, and a set of pairs $X \subseteq P \times P$ that is lexico-consistent.
SOLUTION : Aspect order α such that $E_X(\alpha) = 0$.
MEASURE : $\|\alpha\|$

Theorem 8 *MIN SET COVER is AP-reducible to MIN ASPECTS TO EXPLAIN.*

Proof. We again give a reduction from MIN SET COVER. Given a collection C of subsets of some set S, we construct a MIN ASPECTS TO EXPLAIN instance as follows. Let $C_1, C_2, \ldots, C_n$ be the elements of C and let $x_1, x_2, \ldots, x_m$ be the elements of S. Associate an aspect a_i with each element C_i of C. Associate a profile P_i with each element x_i of S and have P_i contain aspect a_j if and only if $x_i \in C_j$. Create a profile V that contains no aspects. Finally, let $X = \{(P_i, V) : i = 1, \ldots, m\}$. Note that this MIN ASPECTS TO EXPLAIN instance is consistent (assuming $\bigcup_i C_i = S$ and all C_i are nonempty) because any aspect order containing all aspects will differentiate all pairs of profiles in X correctly.

As an example of the transformation, the MIN SET COVER instance

	C_1	C_2	C_3	C_4
x_1	1	0	1	1
x_2	0	1	1	0
x_3	0	1	0	1
x_4	1	0	0	0
x_5	0	0	1	0
x_6	1	0	0	1
x_7	1	1	0	0

becomes

	a_1	a_2	a_3	a_4
P_1	1	0	1	1
P_2	0	1	1	0
P_3	0	1	0	1
P_4	1	0	0	0
P_5	0	0	1	0
P_6	1	0	0	1
P_7	1	1	0	0
V	0	0	0	0

Now suppose we have an r-approximate algorithm for MIN ASPECTS, i.e., the performance ratio is guaranteed to satisfy

$$\frac{|\alpha|}{k^*} \leq r,$$

where k^* is the size of the minimum set cover of the original problem. The best MIN ASPECTS can do is k^* by letting α be a minimum set cover in the original MIN SET COVER instance, because $|\alpha| < k^*$ would mean that at least one element $x_j \in S$ would be uncovered and (P_j, V) would not be differentiated.

Note that the approximate solution α correctly differentiates all pairs in X since α is a feasible solution for MIN ASPECTS TO EXPLAIN. Thus the C_j corresponding to the a_j in α form a set cover C' for the original problem. The performance ratio with respect to MIN SET COVER is

$$\frac{|C'|}{k^*} = \frac{|\alpha|}{k^*} \leq r.$$

Therefore MIN SET COVER is AP-reducible to MIN ASPECTS. □

3.3.5 Minimum distance to a specified order given lexico-consistent

Suppose a set of data is lexico-consistent, i.e., there exists some aspect order that induces no errors with respect to the data. Furthermore, suppose that the research has an idea of what the aspect should have looked like ahead of time, e.g., from self-explicated questions at the beginning of a conjoint analysis survey. Then it might be desirable to find the lexico-consistent aspect order that is closest in some sense to the specified order. We call this problem MIN CONSISTENT DISTANCE, where the distance between two aspect orders is defined as the sum of the absolute differences in aspect position over all aspects.

MIN CONSISTENT DISTANCE	
INSTANCE	: Set of aspects A, an aspect order α (not necessarily lexico-consistent), a set of profiles P, and a set of pairs $X \subseteq P \times P$ that is lexico-consistent.
SOLUTION	: Aspect order α^* such that all pairs in X are differentiated correctly.
MEASURE	: $d(\alpha, \alpha^*) = \sum_i \lvert I_\alpha(a_i) - I_{\alpha^*}(a_i) \rvert$

Theorem 9 *MIN ASPECT POSITION is AP-reducible to MIN CONSISTENT DISTANCE.*

Proof. We give a reduction from MIN ASPECT POSITION. Suppose we have an instance of MIN ASPECT POSITION with aspects A, profiles P, and partial order on profiles X. Let $a_1, a_2, \ldots, a_{n-1}, q$ denote the aspects in A, with q being the special aspect. We construct an instance of MIN CONSISTENT DISTANCE as follows. Let $x_1, x_2, \ldots, x_N$ be additional (dummy) aspects, with $N = n^2$ to ensure that

$$N > \max_{\gamma_1, \gamma_2} d(\gamma_1, \gamma_2),$$

where γ_1 and γ_2 are any permutations of n aspects. Let the desired aspect order for MIN CONSISTENT DISTANCE be

$$\alpha' = (q, x_1, x_2, \ldots, x_N, a_1, a_2, \ldots, a_{n-1})$$

Construct new profiles $P_q = \{q\}$ and $P_i = \{x_i\}$ for all $i = 1, \ldots, N$, and let the set of profiles be

$$P' = P \cup \{P_q\} \cup \{P_i : i = 1, \ldots, N\}$$

Finally, let the partial order on profiles be

$$X' = X \cup \{(P_q, P_i) : i = 1, \ldots, N\}$$

Note that the new pairs in X' force aspect q to come before aspects x_i in any consistent aspect order for the MIN CONSISTENT DISTANCE instance.

In order to analyze the optimal cost for MIN CONSISTENT DISTANCE, consider a consistent aspect order α^* (of the aspects in A) that minimizes the position of aspect q and also minimizes the distance from α^* to $(q, a_1, a_2, \ldots, a_{n-1})$ as a secondary objective. We argue that an optimal aspect order for MIN DISTANCE is β^*, where $x_1, x_2, \ldots, x_N$ immediately follow q, while all other aspects are in the same relative order as in α^*. The total cost is given by

$$cost(\beta^*) = d_1 + d_2,$$

where

$$\begin{aligned} d_1 &= d(\alpha, \alpha^*), \text{ and} \\ d_2 &= (N+1)(I_{\alpha^*}(q) - 1) \end{aligned}$$

The second component of the cost, d_2, is caused by q and x_i shifting to the right due to $(I_{\alpha^*}(q) - 1)$ aspects that must appear before q in α^*.

Now suppose another aspect were shifted before q. This change would increase d_2 by $(N+1)$ while it could only decrease d_1 by at most n^2. So this change would worsen the cost. Furthermore, suppose that the permutation of the aspects (other than q and x_i) were changed. The value of d_2 stays the same, while d_1 becomes worse. Thus β^* achieves the optimal cost for MIN CONSISTENT DISTANCE.

Now suppose we have an r-approximation algorithm for MIN CONSISTENT DISTANCE. Given an approximate solution β, transform it to β' as follows. First reorder the x_i so that they are in increasing order by index. Second, shift all x_i to appear immediately after q (if not already there). This transformation can only increase the quality of the approximate solution, since inserting an x_i at a position k slots to the left of its original position causes at most k aspects (from A) to shift to the right.

We now have

$$\frac{d(\alpha, \beta' \setminus \{x_1, \ldots, x_N\}) + (N+1)(I_{\beta'}(q) - 1)}{d(\alpha, \alpha^*) + (N+1)(I_{\alpha^*}(q) - 1)} \leq r$$

It follows that

$$\frac{d(\alpha, \alpha^*) + (N+1)(I_{\beta'}(q) - 1)}{d(\alpha, \alpha^*) + (N+1)(I_{\alpha^*}(q) - 1)} \leq r$$

since $d(\alpha, \alpha^*) \leq dist(\alpha, \beta' \setminus \{x_1, \ldots, x_N\})$ by the definition of α^*. Finally,

$$\frac{d(\alpha, \alpha^*)/(N+1) - 1 + I_{\beta'}(q)}{d(\alpha, \alpha^*)/(N+1) - 1 + I_{\alpha^*}(q)} \leq r \Rightarrow$$

$$\frac{I_{\beta'}(q)}{I_{\alpha^*}(q)} < r,$$

since $d(\alpha, \alpha^*)/(N+1) - 1 < 0$ by definition of N.

The last ratio is precisely the performance ratio of using $\beta' \setminus \{x_1, \ldots, x_N\}$ as an approximate solution for MIN ASPECT POSITION. Thus, MIN ASPECT POSITION is AP-reducible to MIN CONSISTENT DISTANCE, and it follows from Theorem 7 and the transitivity of AP-reducibility that MIN SET COVER is also AP-reducible to MIN CONSISTENT DISTANCE. □

3.3.6 Minimum distance to a specified order when data not lexico-consistent

Suppose a set of data is not lexico-consistent, i.e., there are no aspect orders that induce zero errors with respect to the data. Furthermore, suppose that the researcher has an idea of what the aspect order should look like ahead of time, e.g., from self-explicated questions at the beginning of a conjoint analysis survey. Then it might be desirable to find the minimum error aspect order (i.e., that has induces that same number of errors as the optimal solution of MIN LEX ERRORS) that is also closest in some sense to the specified order. We call this problem MIN ERROR DISTANCE, where the distance between two aspect orders is defined as the sum of the absolute differences in aspect position over all aspects.

MIN ERROR DISTANCE	
INSTANCE	: Set of aspects A, aspect order α, set of profiles P, set of pairs $X \subseteq P \times P$ that is not lexico-consistent.
SOLUTION	: Aspect order α' such that $E_X(\alpha')$ equals minimum number of errors according to MIN LEX ERRORS
MEASURE	: $d(\alpha, \alpha') = \sum_i \lvert I_\alpha(a_i) - I_{\alpha'}(a_i)\rvert$

Theorem 10 *MIN ERROR DISTANCE is not in APX (unless P = NP).*

Proof. This result follows immediately from Theorem 5. If it were possible to approximate MIN ERROR DISTANCE with some constant factor approximation scheme, then we could use it to solve MIN LEX ERRORS exactly. □

3.3.7 Consistency with ties

Suppose that we allowed the possibility of ties between profiles. Then no aspect order corresponding to the lexicographic decision-making process described earlier can possibly be consistent with data that contains ties (since the lexicographic strategy differentiates every pair of profiles given an aspect order).

Here we introduce a more general lexicographic strategy that allows for ties in the *aspect* ordering. Specifically, instead of depending on an ordered subset of aspects, a lex with ties strategy will depend on an ordered collection of subsets of aspects:

$$S_1 \succ S_2 \succ \ldots \succ S_p,$$

where the sets S_i form a partition of A. When the cardinality of each set is 1, the lex with ties strategy is equivalent to the lex strategy. If $|S_i| > 1$, then the aspects in S_i are valued equally. If two profiles are considered equal before applying S_i, then the profile with more aspects from S_i is preferred.

For example, suppose $A = \{a_1, a_2, a_3\}$, $S_1 = \{a_1, a_2\}$, $S_2 = \{a_3\}$, and we have profiles $P_1 = \{a_1\}$, $P_2 = \{a_2\}$, $P_3 = \{a_1, a_2\}$, and $P_4 = \{a_3\}$. Then we have the

following relationships:

$$P_3 \succ P_1 = P_2 \succ P_3.$$

CONSISTENCY WITH TIES	
INSTANCE :	Set of aspects A, set of profiles P, and set of comparisons X
QUESTION :	Is there a partition of aspects $S_1, S_2, \ldots, S_k$ that is consistent with X?

Theorem 11 *CONSISTENCY WITH TIES is NP-complete.*

Proof. We transform 3-DIMENSIONAL MATCHING (3DM) to CONSISTENCY WITH TIES (CWT). Let W, X, Y, and $M \subseteq W \times X \times Y$ be an arbitrary instance of 3DM, with $|W| = |X| = |Y| = q$. Construct a corresponding instance of CWT as follows. Associate an aspect a_i with every element m_i of M. Let $count(e)$ be the number of times that element e (from W, X, or Y) appears in (elements of) M and let

$$N = \max_{e \in W \cup X \cup Y} count(e).$$

Introduce aspects r and $d_1, d_2, \ldots, d_{N-1}$. Next, associate a profile with each element of W, X, and Y. Specifically, let profile W_i correspond to w_i such that

$$W_i = \{a_j : w_i \in m_j\} \cup \{d_1, d_2, \ldots, d_{N-count(w_i)}\}.$$

That is, profile W_i contains all aspects that correspond to elements of M that contain w_i plus enough dummy aspects d_i to ensure that $|W_i| = N$. Profiles X_i and Y_i are similar. Finally, create profiles Z_1, Z_2, and Z_3, where

$$Z_1 = \{r, d_1, d_2, \ldots, d_{N-1}\}$$
$$Z_2 = \{r\}$$
$$Z_3 = \{d_1, d_2, \ldots, d_{N-1}\}$$

Note that $|Z_1| = N$. Given aspects a_i, r, and d_i, and profiles W_i, X_i, Y_i, and Z_i, we desire comparisons: $W_i = Z_1$, $X_i = Z_1$, $Y_i = Z_1$, and $Z_2 \succ Z_3$. The last comparison causes aspect r to be preferred to d_i in any consistent lex-with-ties strategy (i.e., partition of aspects).

As an example, consider a 3DM instance with $W = \{w_1, w_2, w_3\}$, $X = \{x_1, x_2, x_3\}$, and $Y = \{y_1, y_2, y_3\}$, with

$$M = \{(w_1, x_1, y_2), (w_2, x_3, y_1), (w_3, x_2, y_3), (w_1, x_1, y_1), (w_2, x_2, y_1)\}.$$

The CWT instance is shown in the following table:

	a_1	a_2	a_3	a_4	a_5	r	d_1	d_2
W_1	1	0	0	1	0	0	1	0
W_2	0	1	0	0	1	0	1	0
W_3	0	0	1	0	0	0	0	0
X_1	1	0	0	1	0	0	1	0
X_2	0	0	1	0	1	0	1	0
X_3	0	1	0	0	0	0	1	1
Y_1	0	1	0	1	1	0	0	0
Y_2	1	0	0	0	0	0	1	1
Y_3	0	0	1	0	0	0	1	1
Z_1	0	0	0	0	0	1	1	1
Z_2	0	0	0	0	0	1	0	0
Z_3	0	0	0	0	0	0	1	1

First we show that if a matching exists, then the CWT instance is consistent. (This direction of the proof is easy since it is what drove the construction.) Suppose a matching M' exists. Then the aspect partition $S_1 \succ S_2$ with

$$S_1 = \{a_i : m_i \in M'\} \cup r$$
$$S_2 = \{a_i : m_i \notin M'\} \cup \{d_1, d_2, \ldots, d_{N-1}\}$$

is a consistent lex-with-ties strategy. Under this strategy, all profiles W_i, X_i, Y_i

contain exactly one element from S_1, as does profile Z_1 (aspect r). For example, profile W_i contains exactly one aspect from S_1 since w_i occurs in exactly one element of M'. Furthermore, all profiles W_i, X_i, Y_i, and Z_1 contain $N-1$ aspects from S_2 by construction. Thus all comparisons (x) are satisfied.

Next we show that if the CWT instance is constistent, then a matching exists. Consider the consistent aspect partition

$$S_1 \succ S_2 \succ \ldots \succ S_p$$

Since $Z_2 \succ Z_3$, it must be the case that $r \in S_i$ and $d_k \in S_j$ such that $i < j$. Next, consider the set S' that contains r. Because profile Z_1 must contain exactly one aspect from S', profiles W_i, X_i, Y_i must also contain exactly one aspect from S'. This implies that S' contains aspects corresponding to a matching M', otherwise at least one comparison from (x) would be violated.

Therefore the 3DM instance is a "yes" instance if and only if the CWT instance is a "yes" instance, and CWT is NP-complete. □

Chapter 4

Algorithms

In this chapter, we present solution methods for the problems described in Chapter 3. The problems exhibiting a greedoid language structure (e.g., LEX CONSISTENCY) can be solved with a greedy algorithm over an appropriate objective function. We make use of some additional structure to show that each NP-hard problem (e.g., MIN LEX ERRORS) can be solved with dynamic programming. Since the dynamic programming based algorithms have exponential worst case time complexity, we analyze several heuristics including a greedy heuristic and insertion based local search. We also present several implementation enhancements to the dynamic programming algorithm that provide significant speedup in practice.

4.1 Greedy Algorithms

In each of the decision problems shown to have a greedoid language structure, the decision question can be rephrased as: "Is there a word of length n that is contained in the language?" (where n is the size of the alphabet). The corresponding objective function we are trying to maximize over words in the language is the length of the word. Equivalently, we are maximizing a linear function over the letters where the coefficient corresponding to (the inclusion of) each letter is equal to 1. Recall that this objective function, $W(\alpha)$, was shown to be compatible in Chapter 3.

Let $c(\alpha, x)$ denote the *new* errors directly caused by aspect x occuring after aspect

Algorithm 1 Greedy Algorithm for LEX CONSISTENCY

$\alpha \Leftarrow$ empty word
while $|\alpha| < n$ **do**
  if there exists $x \in E \setminus \alpha$ such that $c(\alpha, x) = 0$ **then**
    $\alpha \Leftarrow \alpha x$
  else
    DONE
  end if
end while

order (or aspect set) α, that is

$$c(\alpha, x) = E_X(\alpha x) - E_X(s) \tag{4.1}$$

Algorithm 1 is the greedy algorithm for LEX CONSISTENCY. (See Boyd [7] for the general form of the greedy algorithm.)

Note that we append any aspect that meets the condition in the if statment because the objective function is simply the length of the aspect order. If Algorithm 1 terminates with $|\alpha| = n$, then the partial order on profiles (or other paired comparison data) is lexico-consistent. Conversely, if $|\alpha| < n$, the partial order on profiles is not lexico-consistent.

The greedy algorithm for BOUNDED ERRORS PER ASPECT is similar. The condition in the if statment is changed to require that $c(\alpha, x) \leq k$. Solving LEX CONSISTENCY or BOUNDED ERRORS PER ASPECT when including unions and intersections of aspects simply requires applying the greedy algorithm over the augmented set of aspects.

4.2 DP Algorithm for MIN LEX ERRORS

The following property of lexicographic preference structures permits us to concentrate on subsets of aspects rather than permutations of aspects.

Lemma 2 *Let α_1 and α_2 be two different permutations of a set of aspects $S \subset A$, and let x be any aspect not in S. Then the number of errors directly caused by x in*

$\alpha_1 x$ is the same as the number of errors directly caused by x in $\alpha_2 x$, i.e., $M_X(\alpha_1 x) = M_X(\alpha_2 x)$.

Proof. Consider a pair of profiles P_i and P_j with $P_i \succ_X P_j$. Either P_i and P_j are differentiated by α_1 (and α_2) or not. If they are already differentiated, then aspect x can not cause a new error regardless of whether if follows α_1 or α_2. On the other hand, suppose P_i and P_j are not yet differentiated by α_1 (and α_2). If aspect x also does not differentiate the profiles, then no error is caused in either case. If aspect x does differentiate the profiles, then

$$x = f_{\alpha_1 x}(P_i, P_j) = f_{\alpha_2 x}(P_i, P_j)$$

and $M_X(\alpha_1 x) = M_X(\alpha_2 x)$ since x differentiates P_i and P_j the same way in both cases. □

Lemma 2 allows us to write the following dynamic programming recursion:

$$J(S) = \min_{x \in S} \{J(S \setminus x) + c(S \setminus x, x)\} \tag{4.2}$$

Translating recursion (4.2) directly into pseudocode leads to the implementation given in Algorithm 2. A table is maintained that corresponds to the optimal values and policies for each possible state. The table contains 2^n entries—one for every possible subset of aspects. Each entry keeps track of the lowest number of errors achievable for some optimal arrangement of those aspects (the optimal value), plus the aspect that occurs in the last position in that optimal arrangement (the optimal policy). The algorithm correctly fills in the table because by the time it is computing table entries for subsets of size k, it has already finished computing optimal values and policies for subsets of size $k - 1$.

Algorithm 2 processes $O(2^n)$ states and requires $O(2^n)$ space. It ultimately performs $O(n2^n)$ stage cost computations (i.e., computations of $c()$ in the innermost loop).

Algorithm 2 Naive DP algorithm for MIN LEX ERRORS

```
1: for k = 1 to n do
2:   for all subsets S of length k do
3:     // perform minimization over aspect in last position
4:     for all i ∈ S do
5:       if T(S \ i).cost + c(S \ i, i) < T(S).cost then
6:         T(S).cost ⇐ T(S \ i).cost + c(S \ i, i)
7:         T(S).aspect ⇐ i
8:       end if
9:     end for
10:  end for
11: end for
```

n	2^n	$n2^n$	$n!$
1	2	2	1
2	4	8	2
3	8	24	6
4	16	64	24
5	32	160	120
6	64	384	720
7	128	896	5,040
8	256	2,048	40,320
9	512	4,608	362,880
10	1,024	10,240	3,628,800
11	2,048	22,528	39,916,800
12	4,096	49,152	479,001,600
13	8,192	106,496	6,227,020,800
14	16,384	229,376	87,178,291,200
15	32,768	491,520	1,307,674,368,000
16	65,536	1,048,576	20,922,789,888,000

Table 4.1: Exponential vs factorial running time

Even though the naive algorithm finds an optimal solution to MIN LEX ERRORS much faster than the $O(n!)$ method presented in Martignon and Schmitt [46] (see Table 4.2, Algorithm 2 can still only be applied to problems of moderate size. However, there are several "tricks", some standard to dynamic programming and some specific to our problem, that substantially speed up the practical running time.

Bounding. Suppose we know that a particular solution for an 6-aspect problem (e.g., provided by the backward insertion heuristic) can achieve as few as 2 errors. Furthermore, suppose the best arrangement of a subset of aspects $S = \{a, b, c\}$ already causes more than 2 errors. Then having aspects d, e, or f in the 4th position after $\{a, b, c\}$ is also guaranteed to cause more than 2 errors. Thus, the stage cost computations $c(S, d)$, $c(S, e)$, and $c(S, f)$ are not necessary (even though they are computed in Algorithm 2). Keeping track of the best known solution can avoid many unnecessary computations.

Differentiation. Once a set of aspects has a certain size, it becomes likely that all pairs of profiles are already differentiated. In a balanced design, a given aspect would be expected to differentiate about half of the pairs of profiles (since the probability of both profiles in a pair being differeniated by a given aspect is 0.5). If the number of undifferentiated pairs halves each time an aspect is added to an order, then we would need approximately

$$\log_2 \binom{m}{2} = \log_2[m(m-1)/2]$$

aspects to differentiate all pairs of profiles.

For example, suppose we have a problem with 32 profiles and 20 aspects. (The experimental design is given in the appendix.) The expected number of aspects required to differentiate each possible pair of profiles is 8.42 with a standard deviation of 1.18 (1000 trials over uniform random aspect orders were conducted). This result matches up well with $\log_2(32 \cdot 31/2) = 8.95$. Table 4.2 shows the fraction of pairs differentiated for each number of aspects as well as the fraction of aspect orders of each length that differentiated all pairs. Repeating this analysis for a 64×40 experimental

k	mean(fd)	stddev(fd)	frac(fd = 1)
1	0.5140	0.001452	0
2	0.7714	0.001601	0
3	0.8962	0.006640	0
4	0.9547	0.006672	0
5	0.9812	0.005228	0
6	0.9924	0.003681	0.016
7	0.9972	0.002371	0.253
8	0.9989	0.001442	0.589
9	0.9996	0.0008291	0.832
10	0.9999	0.0004226	0.954
11	1	0.0002452	0.985
12	1	0.0001103	0.997
13	1	0	1
⋮	⋮	⋮	⋮
20	1	0	1

Table 4.2: Differentiation behavior in 32×20 design

design (given in the appendix) results in an expected number of aspects of 10.84 with a standard deviation of 1.45. This matches up well with $\log_2(64 \cdot 63/2) = 10.98$. Table 4.2 presents more detailed information.

The key observation is that if a set of aspects has already differentiated all pairs of profiles, then appending additional aspects after it has no effect on the number of errors. For example, suppose in a 20 aspect problem that $S = \{a, b, c, d, e\}$ totally differentiates all pairs of profiles. Then computing the stage costs

$$c(S, f), c(S, g), \ldots, c(S, t)$$

is unnecessary. Furthermore, the savings multiply since computing

$$\begin{gathered} c(S \cup f, g), c(S \cup f, h), \ldots, c(S \cup f, t); \\ c(S \cup g, f), c(S \cup g, h), \ldots, c(S \cup g, t); \\ \vdots \\ c(S \cup t, f), c(S \cup t, g), \ldots, c(S \cup t, s); \end{gathered}$$

k	mean(fd)	stddev(fd)	frac(fd = 1)
1	0.5072	0.001089	0
2	0.7609	0.000895	0
3	0.8858	0.002802	0
4	0.9462	0.003203	0
5	0.9753	0.002600	0
6	0.9888	0.002084	0
7	0.9951	0.001501	0
8	0.9978	0.0009794	0.010
9	0.9991	0.0006565	0.144
10	0.9996	0.0004345	0.482
11	0.9998	0.0002682	0.730
12	0.9999	0.0001839	0.877
13	1	0.0001086	0.956
14	1	6.032e-05	0.985
15	1	4.138e-05	0.993
16	1	2.217e-05	0.998
17	1	1.569e-05	0.999
18	1	0	1
⋮	⋮	⋮	⋮
40	1	0	1

Table 4.3: Differentiation behavior in 64×40 design

and so on also becomes unnecessary. For this example, the total number of avoided stage cost computations is $15 \times 14 \times \cdots \times 2 = 15!$.

Dynamic memory allocation. Suppose that the following stage cost computations are unnecessary due to the above considerations:

$$c(\{a, b\}, c), c(\{a, c\}, b), c(\{b, c\}, a).$$

For example, it could be the case that $J(\{a, b\})$, $J(\{a, c\})$, and $J(\{b, c\})$ are all worse than the cost of a known solution. Then $J(\{a, b, c\})$ never has to be computed at all. In the event that many subsets of aspects can be completely ignored, it becomes attractive to only store table entries for those subsets that actually matter.

Combining the preceding observations leads to Algorithm 3. It maintains a FIFO queue of all sets of aspects that have costs (so far) that are less than or equal to the cost of the best known solution. The queue also contains only sets that do not completely differentiate the profiles. These two properties ensures that no unnecessary stage cost computations are performed with respect to bounding and differentation.

Another property of the queue is that the sizes of the sets it contains are nondecreasing. This guarantees that the optimal value for a set of size k is already known before that set is used for stage cost computations on sets of size $k + 1$.

Additionally, the table is designed to contain only those sets that are necessary. The table can be implemented efficiently as a hash table.

Greedoid Initialization. One further tweak is possible, capitalizing on the greedoid structure of the problem. First we prove the following lemma.

Lemma 3 *Suppose that β is an ordered subset of aspects that is lexico-consistent with the preferences in X. Then there is an optimal (full) ordering of aspects that begins with the order β.*

Algorithm 3 Enhanced DP algorithm for MIN LEX ERRORS

```
1: initialize T
2: initialize Q
3: bestSoFar ⇐ cost of heuristic solution
4:
5: // add singleton sets to Q
6: for all aspects i do
7:   if E_X(i) < bestSoFar then
8:     add {i} to T
9:     add {i} to Q
10:  end if
11: end for
12:
13: while Q not empty do
14:   remove first set S from Q
15:   for all i ∈ S do
16:     if T(S).cost + c(S,i) < bestSoFar then
17:       if (S ∪ i) ∈ T then
18:         // update existing table entry if necessary
19:         if T(S).cost + c(S,i) < T(S ∪ i) then
20:           T(S ∪ i).cost ⇐ T(S).cost + c(S,i)
21:           T(S ∪ i).aspect ⇐ i
22:         end if
23:       else
24:         // create new table entry for this set of aspects
25:         add (S ∪ i) to T
26:
27:         // only add to Q if this set will need further processing/expansion
28:         if (S ∪ i) does not totally differentiate P then
29:           add (S ∪ i) to Q
30:         end if
31:       end if
32:
33:       // update bestSoFar if necessary
34:       if (S ∪ i) totally differentiates P and T(S ∪ i).cost < bestSoFar then
35:         bestSoFar ⇐ T(S ∪ i).cost
36:       end if
37:     end if
38:   end for
39: end while
```

Proof. Let α be a solution to MIN LEX ERRORS (with $|\alpha| = |A|$) and suppose there exists a subset of aspects β such that $E_X(\beta) = 0$. Consider the aspect order

$$\gamma = (\beta, (\alpha \setminus \beta))$$

and any aspect $x \in \alpha \setminus \beta$. The number of errors caused directly by x in γ must be less than or equal to the number of errors causes directly by x in α because the set of profiles differentiated by $\gamma^{(I_\gamma(x))}$ is a superset of the profiles differentiated by $\alpha^{(I_\alpha(x))}$. (This is because the set of aspects preceding x in γ is a superset of the set of aspects preceding x in α.) Since all $x \in \alpha \setminus \beta$ cause at most the same number of errors as x in α, $E_X(\gamma) \leq E_X(\alpha)$ which implies that $E_X(\gamma) = E_X(\alpha)$. □

The significance of Lemma 3 is that the greedy algorithm can be run first to find the longest (partial) aspect order that induces no errors (if one exists). Then the problem size can be reduced by removing those aspects and the profiles already differentiated by them.

4.3 Other DP Recursions

The basic property in Lemma 2 extends for most/all of the other hard problems from the previous chapter. What follows are the appropriate modifications to the property (stated without proof) and the resulting DP recursions.

4.3.1 Min Weighted Errors

Lemma 4 *Let α and β be two different permutations of a subset S of aspects, and let x be any aspect not in S. Then the number of weighted errors directly caused by x in αx is the same as the number of errors directly caused by x in βx.*

This property permits the following DP recursion:

$$J(S) = \min_{i \in S}\{J(S \setminus i) + c(S \setminus i, i)\},$$

where

$$c(T,x) = \sum_{(i,j)\in X_T^0 \cap X_{(T,x)}^-} w_{ij}$$

4.3.2 Min aspect position

Lemma 5 *The position of aspect x in α does not change if the permutation of the aspects preceding x is modified.*

Consider the following DP recursion:

$$J(S) = \min_{x\in S} \{J(S \setminus x) + c(S \setminus x, x)\},$$

where

$$c(T,x) = (n+1)\cdot M_X(T,x) + \begin{cases} |T|+1 & \text{if } x = q \\ 0 & \text{otherwise} \end{cases}$$

The first term of the stage cost $c(\cdot,\cdot)$ is incrementally computing the number of errors multiplied by a constant. The second term simply records the position of q. The constant in the first term is chosen to be $n+1$ so that having aspect q earlier in the order can never compensate for having even one more error. (The constant $n+1$ is an upperbound on the position of q.)

Thus, for lexico-consistent data, the DP recursion will favor all lexico-consistent solutions over non lexico-consistent ones, and then select the lexico-consistent solution that allows q to appear earliest in the order. Similarly, for data that is not lexico-consistent, the DP recursion will favor all subsets that achieve the minimum error (i.e., the optimal objective function value of MIN LEX ERRORS), and then select the minimum error solution that allows q to appear earliest in the order.

4.3.3 Min error order closest to a specified order

Lemma 6 *Let $\alpha = \gamma x \delta$, with $|\gamma| = k$, let γ' be a different permutation of γ, and let β be some other aspect order. Then the contribution to $d(\beta,\alpha)$ caused by x is the same as the contribution to $d(\beta,\gamma' x\delta)$ caused by x.*

Note that in each case, the contribution to total distance caused by x is $|I_\beta(x) - (k+1)|$. Now consider the following DP recursion:

$$J(S) = \min_{x \in S} \{J(S \setminus x) + c(S \setminus x, x)\}, \tag{4.3}$$

where

$$c(T, x) = n^2 \cdot M_X(T, x) + [I_\beta(x) - (|T| + 1)].$$

The first term of the stage cost $c(\cdot, \cdot)$ is incrementally computing the number of errors multiplied by a constant. The second term is incrementally computing the distance between the solution and β. The constant in the first term is chosen to be n^2 so that having an aspect order closer to β can never compensate for having more errors. (The constant n^2 is an upper bound on total distance.)

Thus, for lexico-consistent data, the DP recursion will favor all lexico-consistent solutions over non lexico-consistent ones, and then select the lexico-consistent solution that is closest to β. Similarly, for data that is not lexico-consistent, the DP recursion will favor all subsets that achieve the minimum error (i.e., the optimal objective function value of MIN LEX ERRORS), and then select the minimum error solution that is closest to β.

4.3.4 Minimum number of aspects necessary to explain lexico-consistent data

Lemma 7 *The number of new (correct) differentiations directly caused by x in aspect order α does not depend on the permutation of the aspects preceding x in α.*

Consider the following DP recursion:

$$J(S) = \min_{x \in S} \{J(S \setminus x) + c(S \setminus x, x)\},$$

where

$$c(T,x) = \begin{cases} 0 & \text{if } M_X^-(Tx) = 0 \text{ and } M_X^+(Tx) = 0 \\ 1 & \text{if } M_X^-(Tx) = 0 \text{ and } M_X^+(Tx) > 0 \\ \infty & \text{if } M_X^-(Tx) > 0 \end{cases}$$

The third condition of the stage cost function guarantees that any solution with no errors is favored over any solution with errors. The first and second conditions serve to count the number of aspects that actually (correctly) differentiate at least one pair of profiles. Since any aspect that differentiates no profiles at its current position in an aspect order can be moved to a later position in the order without affecting the total number of errors, the DP recursion computes the minimum number of aspects required to differentiate the data.

4.4 Greedy Heuristic

Kohli and Jedidi [39] present a greedy heuristic for minimizing the number of errors with respect to a set of paired comparisons of profiles. Algorithm 4 provides essentially the same algorithm adapted to our formulation and notation.

Algorithm 4 Greedy heuristic for MIN LEX ERRORS

$\alpha \Leftarrow$ empty word
while $|\alpha| < n$ **do**
  $x^* \Leftarrow \arg\min_{x \in E \setminus \alpha} c(\alpha, x)$
  $\alpha \Leftarrow \alpha x^*$
end while

Note that if X is lexico-consistent, then applying Algorithm 4 will find an aspect order α that is consistent with X, i.e., with $E_X(\alpha) = 0$.

4.5 Insertion Heuristics

In the field of combinatorial/discrete optimization, numerous local improvement or neighborhood search heuristics have been developed for solving hard optimization

problems. Here we consider two heuristics based on insertion due to the efficiency of searching over the neighborhood of possible moves.

4.5.1 Forward Insertion

For forward insertion, a move $m = (i \rightarrow j)$, with $i < j$, consists of moving the aspect currently in position i to position j and shifting all aspects currently in positions $i+1, i+2, ..., j$ one position to the left. We will use the notation α_m to denote the resulting aspect order after applying move m to α. For example, if $\alpha = (a, b, c, d, e)$ and $m = (2 \rightarrow 4)$, then $\alpha_m = (a, c, d, b, e)$. Algorithm 5 gives the basic outline of the forward insertion heuristic.

Algorithm 5 Forward insertion heuristic for MIN LEX ERRORS

$\alpha \Leftarrow$ random permutation of $1, \ldots, n$
repeat
  $(i^* \rightarrow j^*) \Leftarrow \arg\min_{i<j}\{d = E_X(\alpha_{(i \rightarrow j)}) - E_X(\alpha)\}$
  if $d^* < 0$ **then**
    $\alpha \Leftarrow \alpha_{(i^* \rightarrow j^*)}$
  end if
until $d^* \geq 0$

A naive implementation would simply compute the total number of errors from scratch for each possible forward insertion. This would have an $O(n^3|\Omega|)$ running time if $O(n^2)$ moves were scored, scoring each move required checking each pair in Ω, and checking each pair required iterating over $O(n)$ aspects.

Due to the special nature of forward insertions, however, the change in the number of errors resulting from applying move m, i.e.,

$$E_X(\alpha_m) - E_X(\alpha)$$

can be computed more efficiently in an incremental manner. Suppose we have an existing aspect order α. Consider a pair (P_i, P_j) and let x_1 be the leftmost aspect in α that differentiates P_i and P_j and let x_2 be the second leftmost aspect in α that differentiates them.

Causing a new error. First, suppose that (P_i, P_j) is currently differentiated correctly, i.e., $x_1 \in A^{\succ}(P_i, P_j)$. Then the only way for a new error to be caused by a move is if

(a) $x_2 \in A^{\prec}(P_i, P_j)$, and

(b) x_1 is inserted behind x_2.

In other words, the only moves that will cause a new error are

$$(I_\alpha(x_1) \rightarrow j), \forall j \geq I_\alpha(x_2).$$

We will refer to this as a *move family*, and refer to it as $(I_\alpha(x_1) \rightarrow \geq I_\alpha(x_2))$.

Fixing an existing error. On the other hand, suppose that (P_i, P_j) is currently differentiated incorrectly, i.e., $x_1 \in A^{\prec}(P_i, P_j)$. Then the only way for this error to be fixed by a move is if

(a) $x_2 \in A^{\succ}(P_i, P_j)$, and

(b) x_1 is inserted behind x_2.

In other words, the only moves that will fix the existing error are

$$(I_\alpha(x_1) \rightarrow j), \forall j \geq I_\alpha(x_2).$$

To exploit these properties, we can maintain a min binary heap for each (P_i, P_j) in Ω containing all aspects in $A^{\succ}(P_i, P_j) \cup A^{\prec}(P_i, P_j)$ sorted by their current positions in α. This allows querying for the first and second minimal elements (corresponding to the leftmost and second leftmost differentiating aspects in α) in constant time (see Cormen et al [11].

For example, suppose

$$P_1 = (100101)$$
$$P_2 = (001110)$$

and $P_1 \succ P_2$, then the heap would include aspects a, c, e, and f. The pair is currently differentiated correctly since $x_1 = a \in A^{\succ}(P_1, P_2)$. However, applying any move from

the move family

$$(I_\alpha(x_1) \rightarrow\geq I_\alpha(x_2)) = (1 \rightarrow\geq 3)$$

would cause a new error.

There are at most $|\Omega|$ move families that result from examining all heaps. Each move family mf has a value associated with it ($mf.delta$) that corresponds to the total change in the number of errors that (any move in) the move family would cause. For example, if a total of three pairs in Ω are affected by the same move family, and two existing errors would be fixed while one new error would be introduced, the combined effect of that move family over all pairs would be -1 (i.e., the total number of errors would decrease by one).

Determining the best move over all move families. Given a list of all relevant move families, it is necessary to compute the best overall move. Suppose all move families that involved moving $\alpha(1)$ to another location were

$$(1 \rightarrow\geq 3).delta = +2$$
$$(1 \rightarrow\geq 5).delta = -3$$
$$(1 \rightarrow\geq 9).delta = +1$$

Remember that moving $\alpha(1)$ to position 3 *or later* causes two new errors. Thus, moving $\alpha(1)$ to position 5 results in a net effect of removing one error. And moving $\alpha(1)$ to position 9 results in a net effect of not changing the total number of errors. If we restricted ourselves to finding the best insertion point for $\alpha(1)$, it would be 5 (or 6, ..., 8).

The important thing to note is that finding the optimal insertion point for $\alpha(1)$ did not necessarily require considering all $O(n)$ insertion points. Likewise, finding the best move over all origins (that occur in at least one move family) only requires considering at most $|\Omega|$ move families rather than $O(n^2)$ possible moves. The only requirement is that the move families be sorted first by origin, then by destination. A radix sort can accomplish this in $O(|\Omega|)$ time.

Updating after a move. Once a best move $(i^* \rightarrow j^*)$ has been found, the aspect

order and heaps must be updated. Because the relative positions of all aspects other than $\alpha(i^*)$ stay the same, the relative positions of those aspects in each heap remain the same. Only the relative position of aspect $\alpha(i^*)$ to the other aspects changes. Specifically, its position increases and the heap property can become violated. In order to restore the heap property, $\alpha(i^*)$ can be swapped (downward) with its smallest child as needed. Because the depth of each heap is $O(\log n)$, updating all heaps requires $O(\log n|\Omega|)$ time.

Overall algorithm and complexity. The forward insertion heuristic with a heap-based implementation first initializes all heaps which takes $O(n \log n|\Omega|)$ time ($O(n)$ time for finding the elements that need to go in each heap and $O(n \log n)$ time for constructing each heap). Each iteration involves finding the best move and updating the heaps after applying that move (if the move is an improvement). Finding the best move (see Algorithm 6) takes $O(|\Omega|)$ time since detecting all relevant move families takes $O(|\Omega|)$ time, the radix sort over origins and destinations takes $O(|\Omega|)$ time, and finding the best move over all move families takes $O(|\Omega|)$ time. As mentioned above, updating the heaps takes $O(\log n|\Omega|)$ time. Thus, each iteration requires $O(\log n|\Omega|)$ time rather than the $O(n^3|\Omega|)$ time complexity of the naive implementation.

4.5.2 Backward Insertion

For backward insertion, a move $m = (j \leftarrow i)$, with $j < i$, consists of moving the aspect currently in position i to position j and shifting all aspects currently in positions $i, i+1, ..., j-1$ one position to the right. For example, if $\alpha = (a, b, c, d, e)$ and $m = (2 \leftarrow 5)$, then $\alpha_m = (a, e, b, c, d)$. Algorithm 7 provides a high level view of the method (which strongly resembles the forward version). The analysis for finding the best backward move is similar to the forward move case, but differs in a critical way.

Causing a new error. Suppose that (P_i, P_j) is currently differentiated correctly, i.e., $x_1 \in A^{\succ}(P_i, P_j)$. Then the only way for a new error to be caused by a backward move is if some aspect $x \in A^{\prec}(P_i, P_j)$ is moved before x_1. Thus, instead of a single relevant move family (as in the forward case), $O(n)$ move families can be relevant for

Algorithm 6 Finding best forward insertion move

```
1: // find all move families that affect number of errors
2: for all (P_i, P_j) ∈ Ω do
3:    x_1 ⇐ heap(P_i, P_j).first
4:    x_2 ⇐ heap(P_i, P_j).second
5:
6:    // see what move family would cause new error
7:    if x_1 ∈ A^≻(P_i, P_j) and x_2 ∈ A^≺(P_i, P_j) then
8:       // update delta for move family
9:       T[I_α(x_1) → I_α(x_2)] ⇐ T[I_α(x_1) → I_α(x_2)] + 1
10:   end if
11:
12:   // see what move family would fix existing error
13:   if x_1 ∈ A^≺(P_i, P_j) and x_2 ∈ A^≻(P_i, P_j) then
14:      // update delta for move family
15:      T[I_α(x_1) → I_α(x_2)] ⇐ T[I_α(x_1) → I_α(x_2)] − 1
16:   end if
17: end for
18:
19: // create sorted list of destinations for each origin (a list of lists)
20: moveList ⇐ radixSort(T)
21:
22: // find best move
23: bestMove.delta ⇐ ∞
24: for all orig in moveList do
25:    currentDelta ⇐ 0
26:    for all dest in moveList(orig) do
27:       currentDelta ⇐ currentDelta + T[orig → dest]
28:       if currentDelta < bestMove.delta then
29:          bestMove.move ⇐ (orig → dest)
30:          bestMove.delta ⇐ currentDelta
31:       end if
32:    end for
33: end for
```

Algorithm 7 Backward insertion heuristic for MIN LEX ERRORS

```
α ⇐ random permutation of 1, ..., n
repeat
   (j* ← i*) ⇐ arg min_{j<i} {d = E_X(α_(j←i)) − E_X(α)}
   if d* < 0 then
      α ⇐ α_(j*←i*)
   end if
until d* ≥ 0
```

a given pair of profiles.

Fixing an existing error. On the other hand, suppose that (P_i, P_j) is currently differentiated incorrectly, i.e., $x_1 \in A^{\prec}(P_i, P_j)$. Then the only way for this error to be fixed by a move is if some aspect $x \in A^{\succ}(P_i, P_j)$ is moved before x_1. Here again, instead of a single relevant move family (as in the forward case), $O(n)$ move families can be relevant for a given pair of profiles.

Determining the best move over all move families. Once all relevant move families have been computed, the best overall move can be determined in a similar way to the forward insertion case. The only difference is that the result of the radix sort must have the move families sorted by origin, and then by destination in *decreasing* order.

Updating after a move. Once a best move $(j^* \leftarrow i^*)$ has been found, all heaps need to be updated. As in the forward insertion case, we only need to be concerned with the relative positioning of $\alpha(i^*)$. Since the position of $\alpha(i^*)$ has decreased, the heap property may now be violated. However, it can be restored by swapping $\alpha(i^*)$ (upward) with its parent as needed. This is an $O(\log n)$ operation per heap.

Overall algorithm and complexity. The form of the overall algorithm and the complexity analysis is very similar to the forward insertion case. The only difference is that there are $O(n|\Omega|)$ relevant move families. Thus, each iteration takes $O(n|\Omega|)$ time.

4.6 Numerical Results

4.6.1 Comparison of heuristics

In order to compare the performance of the greedy and insertion heuristics, we conducted a series of experiments. We consider two methods for generating partial orders over profiles, each intended to allow varying the degree of lexicography (or degree of "lexico-consistency"). For one set of experiments, partial orders over profiles were generated in the following way:

1. Given $k \geq 0$.

2. Construct a partworth vector w such that each element w_i is uniformly distributed in $[0, 1]$.

3. Raise each element of w to the kth power.

4. Then P_i preferred to P_j if $\sum_{a \in P_i} w_a > \sum_{a \in P_j} w_a$.

Note that this is the same Einhorn generating model discussed in Section 2.3. When $k = 0$, we have the Dawes equal weights model. When $k = 1$, we have uniform random weights. When k is large, the operation of raising each element to the kth power behaves similarly to the L_∞ norm in that it "selects" and exaggerates the largest element. It also exaggerates the second largest element with respect to the remaining elements, and so on. So as k grows, the degree of lexicography grows.

A second method for generating partial orders over profiles is as follows:

1. Given a random lexico-consistent profile order (i.e., a linear ordering of the profiles that is induced by some random aspect order).

2. Given a number of iterations, *flips*.

3. Swap a random profile with its successor (in the linear ordering of the profiles).

4. Perform Step (3) *flips* times.

Here we start with a perfectly lexico-consistent profile order and introduce possible errors by repeatedly swapping neighboring profiles in the order. When *flips* grows large, the profile order approaches a uniform random profile order.

The results of varying k from 1/8 to 64 on designs of size 32×20 and 64×40 (see Appendix A for details) are shown in Tables 4.4 and 4.5. The results of varying *flips* from 2560 down to 0 are given in Tables 4.6 and 4.7. Each heuristic was run 20 times for each parameter setting (i.e., each row of each table), and the results averaged. The columns have the following meanings: cost = number of errors, diff = number of errors - optimal number of errors, time = running time in seconds, and moves =

number of insertions. All the experiments in this chapter were carried out on a laptop with an Intel Centrino 1.6 GHz CPU, 1 GB of memory, and the Fedora Core 4 linux operating system. The algorithms were coded with the Java programming language.

Discussion. Both the greedy heuristic and backward insertion heuristic perform very well over all designs and generation method parameters. The performance of both greedy and backward improves as the degree of lexicography increases (i.e., as k increases and $flips$ decreases). It is also interesting to note that the hardest case to fit was $k = 1/8$ for the 64×40 design. This suggests that a very compensatory profile order is harder to fit than a random one (e.g., $flips = 2560$ for the 64×40 design).

Interestingly, even though the forward insertion heuristic has a slightly better asymptotic running time per iteration, it performs poorly compared to the backward insertion heuristic and greedy heuristic. Furthermore, the performance appears to degrade as the degree of lexicography increases. Wolpert and Macready [70] emphasize that incorporating problem-specific knowledge into an optimization algorithm is very important. For any given problem, some algorithm is best, and which is the best depends heavily on the "landscape" of the objective function.

In the case of finding aspect orders that minimize errors (wrong comparisons), it appears that moving aspects to earlier positions in the order is much more effective than moving aspects forward (later in the order). This makes sense intuitively since moving good/important aspects to the front of the order seems to have the potential for larger gains than simply moving unimportant aspects out of the way. Practically speaking, the backward insertion heuristic appears less susceptible to getting stuck in (poor) local optima and often performs as well or better than the greedy heuristic (which also focuses on the most important aspects first).

4.6.2 Scalability and robustness of the DP algorithm

In order to test the effectiveness of Algorithm 3, we applied the algorithm (seeding it with bounds from the greedy heuristic) over all parameter settings discussed in the previous experiment. The results are given in Tables 4.8, 4.9, 4.10, and 4.11.

	greedy heuristic			backward insertion				forward insertion			
k	cost	diff	time	cost	diff	time	moves	cost	diff	time	moves
1/8	135.60	1.55	0.012	136.50	2.45	0.033	5.80	158.40	24.35	0.041	4.05
1/4	133.10	0.90	0.011	135.10	2.90	0.029	5.55	159.65	27.45	0.040	4.50
1/2	130.60	1.95	0.011	131.65	3.00	0.029	5.45	150.35	21.70	0.049	5.95
1	122.95	3.05	0.012	122.05	2.15	0.030	6.05	152.70	32.80	0.041	4.70
2	100.30	1.20	0.011	99.85	0.75	0.032	6.70	147.60	48.50	0.043	5.05
4	79.15	0.70	0.011	79.25	0.80	0.030	6.25	140.10	61.65	0.040	4.60
8	44.05	0.35	0.011	44.25	0.55	0.030	6.35	165.40	121.70	0.036	3.90
16	14.75	0.00	0.011	14.75	0.00	0.029	6.50	146.80	132.05	0.048	5.80
32	7.15	0.00	0.011	7.35	0.20	0.029	6.35	127.50	120.35	0.052	6.55
64	0.85	0.00	0.011	0.95	0.10	0.029	6.40	147.70	146.85	0.052	6.40

Table 4.4: Comparison of heuristics for 32×20 design as k varies

	greedy heuristic			backward insertion				forward insertion			
k	cost	diff	time	cost	diff	time	moves	cost	diff	time	moves
1/8	718.40	9.75	0.015	720.20	11.55	0.473	7.60	775.15	66.50	0.808	5.95
1/4	705.35	9.70	0.014	706.75	11.10	0.495	8.90	769.55	73.90	0.860	6.90
1/2	683.60	8.15	0.013	684.45	9.00	0.418	7.30	762.35	86.90	0.771	6.05
1	649.50	6.80	0.013	650.60	7.90	0.457	8.15	751.05	108.35	0.832	6.65
2	592.80	4.75	0.013	594.95	6.90	0.469	8.15	711.20	123.15	0.957	7.75
4	483.90	2.15	0.013	483.05	1.30	0.464	8.45	689.55	207.80	1.003	8.15
8	348.20	1.55	0.013	347.85	1.20	0.445	8.35	685.70	339.05	0.961	7.85
16	143.80	0.80	0.013	143.45	0.45	0.437	8.50	683.05	540.05	0.826	6.65
32	88.15	0.00	0.013	88.25	0.10	0.428	8.60	770.85	682.70	0.841	6.70
64	16.50	0.05	0.013	16.75	0.30	0.424	8.70	800.75	784.30	0.908	7.35

Table 4.5: Comparison of heuristics for 64×40 design as k varies

	greedy heuristic			backward insertion				forward insertion			
flips	cost	diff	time	cost	diff	time	moves	cost	diff	time	moves
2560	132.50	2.10	0.011	131.85	1.45	0.031	6.00	169.80	39.40	0.046	5.45
1280	107.50	2.35	0.011	106.45	1.30	0.031	6.50	175.70	70.55	0.043	4.95
640	78.30	1.85	0.012	77.70	1.25	0.031	6.50	160.65	84.20	0.043	4.95
320	56.95	0.95	0.012	57.25	1.25	0.029	6.10	158.15	102.15	0.042	4.90
160	40.60	1.40	0.012	40.55	1.35	0.029	6.10	164.50	125.30	0.047	5.65
80	26.80	0.85	0.011	27.10	1.15	0.030	6.35	156.70	130.75	0.039	4.35
40	17.80	0.95	0.011	17.20	0.35	0.029	6.30	157.85	141.00	0.051	5.85
20	11.75	0.70	0.011	11.85	0.80	0.029	6.40	171.50	160.45	0.045	5.25
10	6.35	0.10	0.012	6.45	0.20	0.029	6.40	123.80	117.55	0.039	4.45
0	0.00	0.00	0.012	0.00	0.00	0.032	6.35	164.25	164.25	0.041	4.05

Table 4.6: Comparison of heuristics for 32×20 design as *flips* varies

	greedy heuristic			backward insertion				forward insertion			
flips	cost	diff	time	cost	diff	time	moves	cost	diff	time	moves
2560	262.75	3.20	0.012	262.35	2.80	0.437	8.40	694.70	435.15	0.938	7.55
1280	175.85	1.45	0.013	176.60	2.20	0.443	8.75	732.30	557.90	0.813	6.45
640	129.90	2.65	0.013	129.60	2.35	0.432	8.55	666.35	539.10	0.910	7.40
320	81.05	1.70	0.013	81.15	1.80	0.447	8.90	679.30	599.95	1.025	8.30
160	57.85	1.60	0.013	57.75	1.50	0.439	9.00	667.70	611.45	0.936	7.50
80	36.65	1.10	0.013	36.80	1.25	0.428	8.80	804.90	769.35	0.831	6.60
40	20.80	0.60	0.013	21.00	0.80	0.429	8.80	833.80	813.60	0.707	5.50
20	13.20	0.40	0.013	13.60	0.80	0.422	8.75	703.85	691.05	0.791	6.30
10	7.75	0.00	0.013	7.85	0.10	0.431	8.80	728.45	720.70	0.803	6.45
0	0.00	0.00	0.014	0.20	0.20	0.469	8.85	755.15	755.15	0.851	6.70

Table 4.7: Comparison of heuristics for 64×40 design as $flips$ varies

k	cost	time	entries	computations
1/8	134.05	0.03	229.85	3789.90
1/4	132.20	0.01	170.20	2907.25
1/2	128.65	0.02	224.80	3674.25
1	119.90	0.02	212.25	3320.35
2	99.10	0.00	61.25	984.20
4	78.45	0.00	30.45	487.90
8	43.70	0.01	12.65	216.60
16	14.75	0.00	2.00	56.20
32	7.15	0.00	0.95	37.15
64	0.85	0.00	0.05	20.95

Table 4.8: DP performance for 32×20 design as k varies

k	cost	time	entries	computations
1/8	708.65	2.34	20414.30	709590.35
1/4	695.65	2.37	25381.40	865289.70
1/2	675.45	1.55	15282.75	523247.95
1	642.70	0.71	6528.30	224503.00
2	588.05	0.26	2242.40	76559.60
4	481.75	0.05	448.15	15101.40
8	346.65	0.01	112.00	3727.20
16	143.00	0.01	49.00	1617.80
32	88.15	0.00	4.45	204.85
64	16.45	0.00	3.15	149.10

Table 4.9: DP performance for 64×40 design as k varies

The columns have the following meanings: cost = optimal number of errors, time = running time in seconds, entries = total number of entries stored in the table, and computations = total number of stage cost computations.

Discussion. The number of table entries and computations decreases as the degree of lexicography increases, i.e., the algorithm is faster when the data is more lexicographic. For all parameter settings, the running time is very reasonable. Running times on the order of hundreths of a second for a 40 aspect problem compare very favorably to the running time of 2 days for 9 aspects reported in Martignon and Hoffrage [45]. In fact, for problem sizes likely to be found in practice, the speeds achieved by Algorithm 3 would enable solving lexicographic inference problems on

flips	cost	time	entries	computations
2560	130.40	0.02	205.95	3190.05
1280	105.15	0.01	135.15	2038.50
640	76.45	0.01	126.85	1879.80
320	56.00	0.01	51.45	780.25
160	39.20	0.01	68.30	1009.70
80	25.95	0.00	27.75	416.20
40	16.85	0.00	29.80	441.80
20	11.05	0.00	17.05	259.00
10	6.25	0.00	5.75	113.30
0	0.00	0.00	0.00	20.00

Table 4.10: DP performance for 32×20 design as *flips* varies

flips	cost	time	entries	computations
2560	259.55	0.07	803.35	26547.90
1280	174.40	0.02	224.65	7467.15
640	127.25	0.04	510.50	16674.35
320	79.35	0.02	192.85	6266.35
160	56.25	0.02	181.30	5902.05
80	35.55	0.01	72.90	2353.00
40	20.20	0.01	33.65	1109.85
20	12.80	0.00	24.60	832.80
10	7.75	0.00	5.70	248.75
0	0.00	0.00	0.00	40.00

Table 4.11: DP performance for 64×40 design as *flips* varies

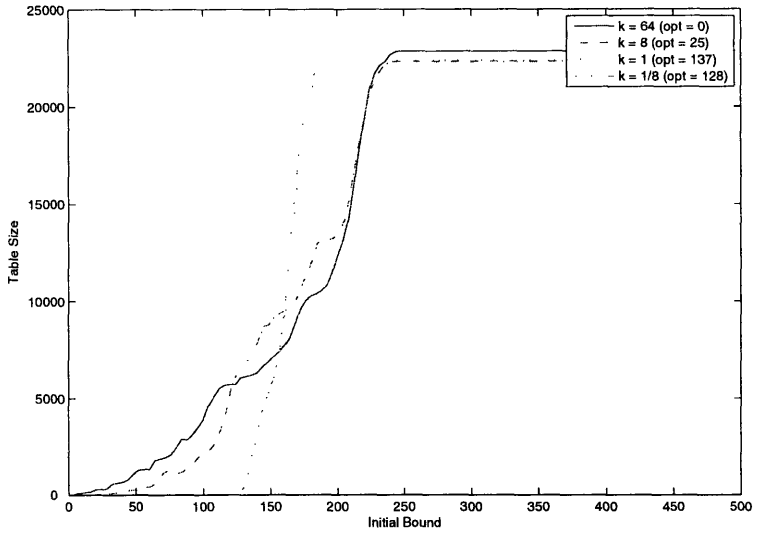

Figure 4-1: Number of table entries vs goodness of initial bound

the fly, say as part of an online survey.

4.6.3 Sensitivity to initial bound

Much of the success of Algorithm 3 can be attributed to the goodness of the initial bound. Here we explore this by generating 4 subjects with k taking on values 64, 8, 1, and 1/8. The results of running the DP with different hypothetical initial bounds are show in Figure 4.6.3. In essence, we are considering how sensitive the running time (or equivalently, the number of table entries) is to the goodness of the initial bound.

Discussion. For each value of k, the total number of table entries is approximately $23,000$ when no initial bound is given. (Note that there can be at most 496 wrong comparisons given 32 profiles.) What Figure 4.6.3 suggests is that having an initial bound (say from the greedy heuristic) that is close to the optimal number of errors can drastically reduce the number of table entries (and stage cost computations) required. Is also appears that the algorithm is less sensitive to the goodness of the initial bound when the optimal number of errors is low (i.e., the data are close to lexicographic).

Chapter 5

Empirical Studies

In this chapter, we report on an empirical study of SmartPhone preferences. We describe the experimental setup and data collection, perform several analyses using noncompensatory, compensatory, and constrained compensatory models, and discuss behavioral and managerial implications of the results. We also analyze an additional dataset in another product category (computers) that was generously provided by another team of researchers. We draw similar, though slightly different conclusions for the second set of data.

5.1 Basic conjoint analysis study

The basic goal of a conjoint analysis study is using stated and/or revealed preference data to learn about the decision-making processes in a given population and context, and to subsequently apply those insights to predict future preferences or behavior.

Datasets for studies of consumer preferences are often collected via a survey or questionnaire in which a subject rates or makes choices between different profiles (e.g., products). Holdout (or validation) questions are usually included in the questionnaire. These extra observations (either choices or ratings) are not used to fit the models, but are instead kept separate for validating the models after estimation. They serve as a proxy for truth and enable the comparison of competing methods.

5.2 Benchmarks

For choice data (rank order, choice-based, etc.), LINMAP and Hierarchical Bayes (HB) are two well-established estimation methods. As discussed in Section XXX, these two methods can be adapted to accomodate q-compensatory constraints. Although HB seems to be considered the most popular method to estimate additive models in conjoint analysis, we will use both HB and LINMAP as compensatory benchmarks.

Since we will be using HB in a choice setting (instead of metric), we will refer to it as hierarchical Bayes ranked logit (HBRL). We use the most-recent version of LINMAP which enforces strict rankings (Srinivasan [62]). Both benchmark methods predict holdouts slightly better than either traditional LINMAP (Srinivasan and Shocker [63]) or analytic-center estimation (Toubia et al [66]).

5.3 SmartPhone study

We invited respondents to complete a web-based questionnaire about SmartPhones. The respondents were students drawn from the undergraduate and graduate programs at two universities. To the best of our knowledge, they were unaware of greedoid methods or the purpose of our study. As an incentive to participate, they were offered a 1-in-10 chance of winning a laptop bag worth $100, yielding a 63% response rate. Pretests in related contexts suggested that SmartPhones were likely to include noncompensatory features and thus represented an interesting category for a first test of greedoid methods.

The survey consisted of six phases. The first three phases are as described in Figures 5-1, 5-2, and 5-3: respondents were introduced to the category and SmartPhone features, indicated which SmartPhones they would consider (in half the cells), and successively chose SmartPhones in order to rank their considered products (or rank all products, depending on cell). Respondents then completed a mini-IQ test to cleanse memory--a task which pretests suggested was engaging and challenging.

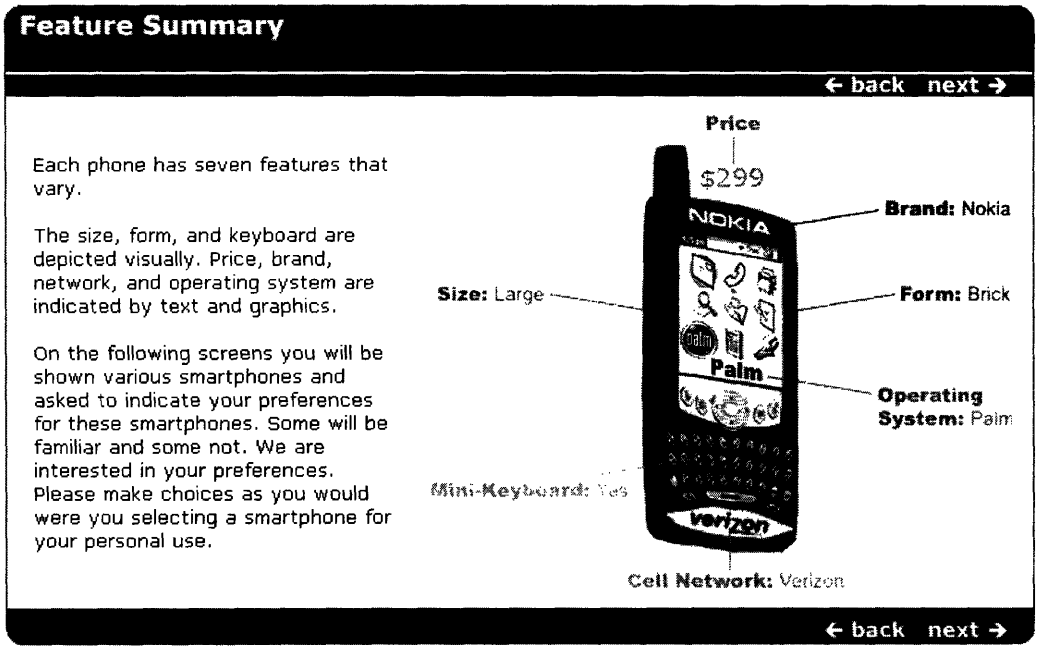

Figure 5-1: SmartPhone Features

Following this filler task, respondents completed a holdout task consisting of two sets of four SmartPhones chosen randomly from a different 32-profile fractional factorial design.15 The final task was a short set of questions about the survey itself—data which we use to compare task difficulty. For the holdout task, in order to avoid unwanted correlation due to common measurement methods, we used a different interface. Respondents used their pointing device to shuffle the profiles into a rank order as one might sort slides in PowerPoint. Pretests suggested that respondents understood this task and found it different from the task in Figure 5-3.

The survey was programmed in PHP and debugged through a series of pretests with 56 respondents chosen from the target population. By the end of the pretests, all technical glitches were removed. Respondents understood the tasks and found them realistic.

5.3.1 Experimental Design

Respondents were assigned randomly to experimental cells. The basic experimental design is a 2×2 design in which respondents complete either a full-rank or a consider-

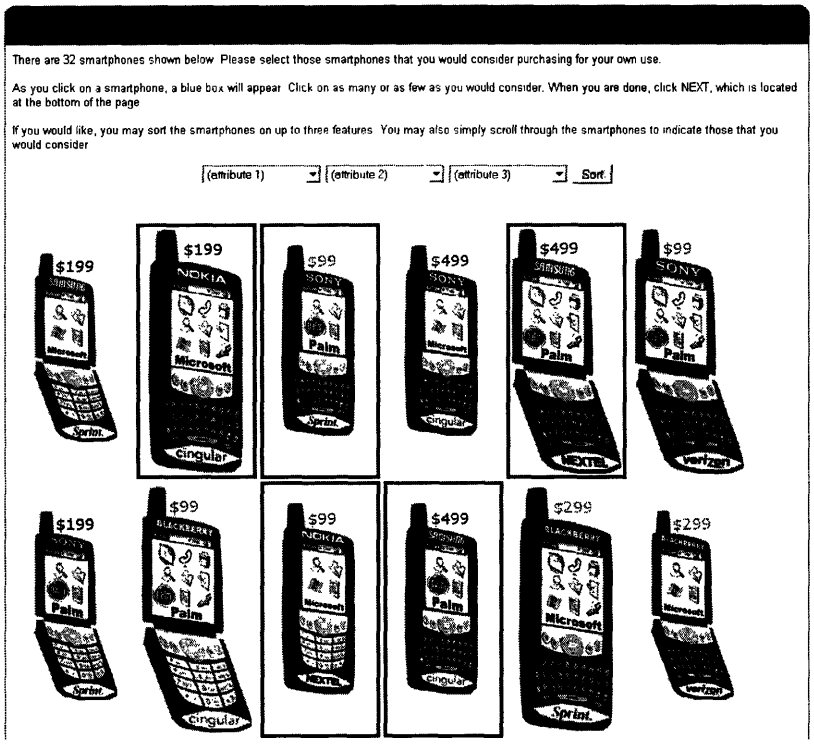

Figure 5-2: SmartPhone Consideration Stage

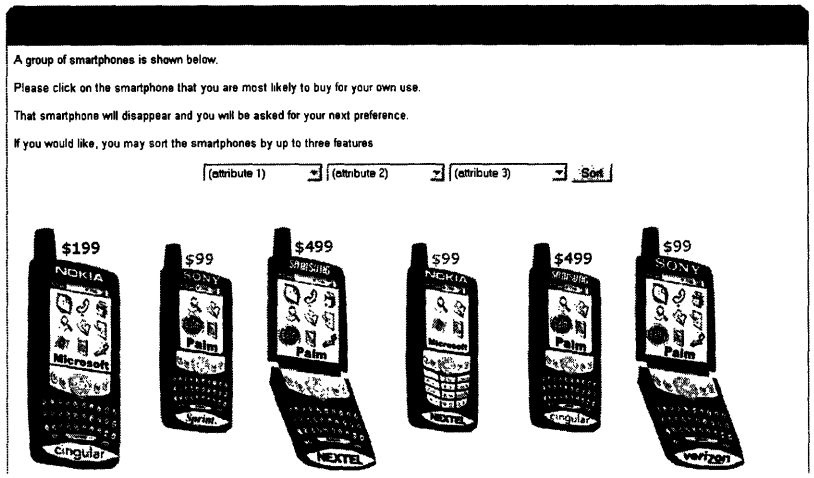

Figure 5-3: SmartPhone Ranking Stage

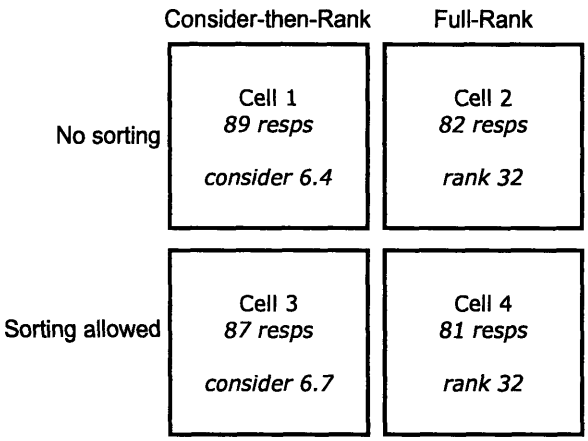

Figure 5-4: SmartPhone Experimental Design (32 Profiles in a $4^3 2^4$ Fractional Factorial Design)

then-rank task and are given the opportunity to presort profiles or not (Figure 5-4). In the consider-then-rank sort cell, respondents could sort prior to consideration and prior to choice. Respondents in the sort cells could re-sort as often as they liked. We also included an additional cell (described below) to test whether the results vary by the number of profiles presented to the respondents. This experimental design enables us to test greedoid methods with different data collection tasks and to illustrate how greedoid methods might be used to explore on how context affects respondents processing strategies.

5.3.2 Task Difficulty

Greedoid methods can be used to analyze any full- or partial-order respondent task. We first examine whether the consider-then-rank task is more natural and easier for respondents than the full-rank task. The results are reported in Figures 5-5 and 5-6. We oriented both axes such that down is better. In the base condition of no sorting, the consider-then-rank task is seen as significantly more enjoyable, accurate, and engaging ($t = 2.2$, $p = 0.03$), saves substantial time (3.75 minutes compared to 8.75 minutes, $t = 2.8$, $p = 0.01$), and appears to increase completion rates (94% vs. 86%, $t = 1.7$, $p = 0.09$). Sorting (as implemented) mitigates these advantages: neither attitudes, time, nor completion rates are significantly different between the full-rank

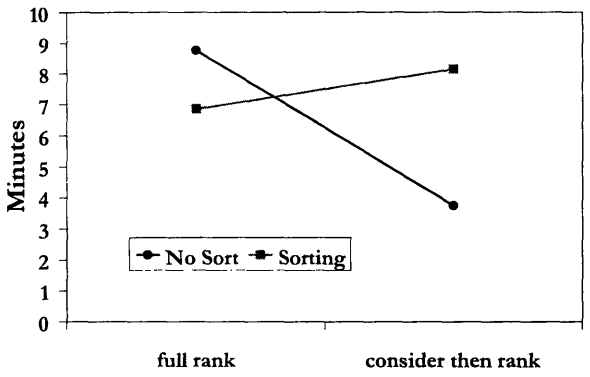

Figure 5-5: Times to complete the tasks (less is better)

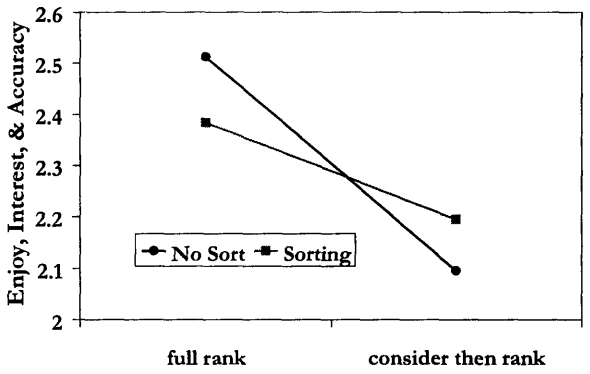

Figure 5-6: Attitudes toward task difficulty (less is better)

and consider-then-rank tasks when respondents can pre-sort profiles.[1] A possible explanation is that sorting made the full rank task easier (though not necessarily more enjoyable), while making the consider-than-rank task more complex.

5.3.3 Predictive Ability

We first compare the most general greedoid method (LBA) to the unconstrained additive models HBRL and LINMAP, as averaged across respondents. Holdout predictions are based on two metrics (see Table 5.1). Hit rate provides fewer observations per respondent (2) and leads to more ties, but is not optimized directly by either greedoid methods or the benchmarks. The percent of violated pairs provides more observations per respondent (12 potential pairs from two sets of four ranked profiles), but is the

[1]For the sorting cells, attitudes ($t = 0.9$, $p = 0.37$), time ($t = 0.4$, $p = 0.70$), and completion rate ($t = 1.1$, $p = 0.26$). Using analysis of variance, there is an interaction between sorting and task for time, but it is not significant ($F = 2.6$, $p = 0.11$). For attitudes only task is significant ($F = 4.9$, $p = 0.03$).

	LBA	HBRL	LINMAP	LBF
Fit (percent pairs)	0.955*	0.871	0.969†	0.826
Holdout (percent pairs)	0.745*	0.743	0.737	0.658
Holdout (hit rate)	0.597**	0.549	0.549	0.481

**LBA significantly better than HBRL, LINMAP, and LBF. *LBA significantly better than LBF. †LINMAP significantly better than LBA and HBRL. Tests at the 0.05 level.

Table 5.1: Comparison of Fit and Prediction for Unconstrained Models

metric optimized by greedoid methods, and, to some extent, by LINMAP. Empirically, the two metrics are significantly correlated (< 0.001 level) for all methods and provide similar comparative qualitative interpretations.[2]

As expected the unconstrained LINMAP, which nests LBA and optimizes a metric similar to the fit metric, provides the best fit. However, LBA fits almost as well. The more interesting comparisons are on the two holdout metrics. For both metrics, LBA is better than both benchmarks and significantly better on hit rates. It appears that, for these data, greedoid methods are more robust than the unconstrained additive models that could, in theory, fit a lexicographic process. This apparent robustness is consistent with predictions by Mitchell [48] and Martignon and Hoffrage [45]. We address the last column of Table 5.1 later in this section.

5.3.4 Comparison to q-compensatory Processes

Following Bröder [9] we examine whether respondents are described better by lexicographic or q-compensatory processes. Three comments are in order. First, this description is paramorphic. We say only that respondents rank (choose, consider) profiles *as if* they were following one or the other process. Second, we have some confidence in the descriptions because LBA predicts better for synthetic respondents who are lexicographic and a constrained additive model (q-compensatory) predicts better for synthetic respondents who are q-compensatory (see Section XXX). Third, for simplicity of exposition, we compare LBA to the HBRL benchmark. This benchmark does slightly better than LINMAP in Table 5.1 and, we will see later, better

[2]For example, correlations between the metrics are 0.70 for LBA, 0.64 for HBRL, and 0.66 for LINMAP.

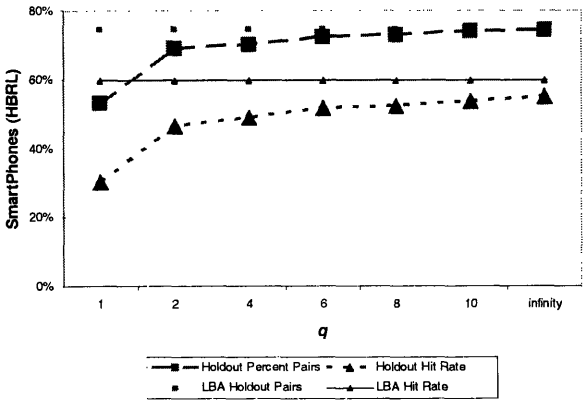

Figure 5-7: Comparison of Holdout Prediction for q-compensatory Models

for a second data set.

Figure 5-7 plots holdout predictions as a function of q. Predictions improve as the models become less constrained (larger q), consistent with a perspective that some aspects are either being processed lexicographically or have large relative partworths. HBRL(q) approaches LBAs holdout percent-pairs-predicted for large q, but falls short on holdout hit rates.

At the level of the individual respondent, comparisons depend upon the choice of q. As an illustration we use $q = 4$. At $q = 4$ the respondent is acting as if he or she is making tradeoffs among aspects by weighing their partworths. Furthermore, the analysis of synthetic data suggests that at $q = 4$ respondents that are truly compensatory are classified as compensatory and respondents who are truly lexicographic are classified as lexicographic.

For holdout percent pairs, LBA predicts better than HBRL(4) for 56% of the respondents, worse for 43% of the respondents, and is tied for 1% of the respondents. On average LBAs predictive ability is about 5 percentage points higher than HBRL(4). The corresponding comparative percentages for hit rates are 46%, 30%, and 24%.[3] On average, LBAs hit rate is about 11 percentage points higher than HBRL(4). Figure 5-8 and 5-9 provides a visual comparison of the distributions of holdout metrics for individual respondents. Positive numbers (darker bars) indicate those respondents for which LBA predicts better than HBRL(4). These percent-

[3]At the level of individual respondents, hit rates are coarser measures than the percent of violated pairs, hence more ties are observed.

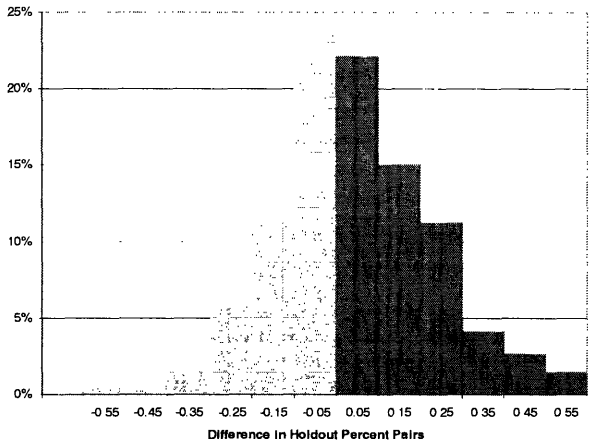

Figure 5-8: Histogram of Comparative Predictive Ability : Percent Pairs

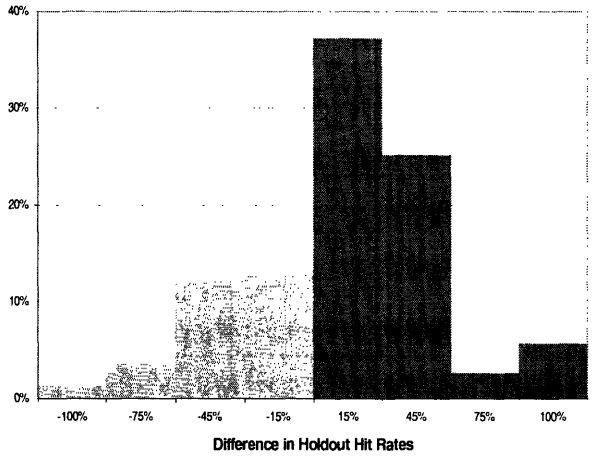

Figure 5-9: Histogram of Comparative Predictive Ability : Hit Rates

ages and Figures 5-8 and 5-9 suggest that greedoid methods are a viable method to complement more-traditional methods to evaluate whether respondents are using compensatory or noncompensatory processes.

5.3.5 Constructed Processes : Full-rank vs Consider-then-rank; Sorting vs Not Sorting

Behavioral researchers hypothesize that consumers construct their decision processes as they make their decisions and, hence, that these decision processes can be influenced by the nature of the decision task. We examine this issue by comparing the influence of task (consider-then-rank vs. full-rank) and the availability of a pre-sorting mechanism (sorting allowed vs. not allowed). Figures 5-10 and 5-11 compare

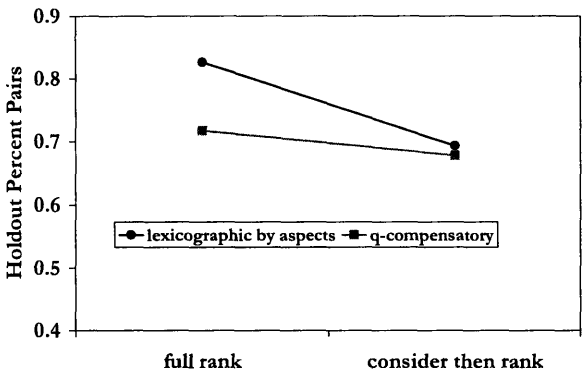

Figure 5-10: Predictive Ability by Experimental Cell, Lexicographic vs. q-Compensatory Processes : No sorting allowed

the predictive ability (holdout violations) for the four cells of our basic experiment. Some insights are:

1. Allowing respondents to presort SmartPhones does not have a significant effect on either LBA or HBRL(4). Task has a significant effect for both LBA and HBRL(4).[4]

2. On average, LBA predicts significantly better than a q-compensatory model in full-rank cells ($t = 6.0$, $p = 0.0$), but not in the consider-then-rank cells ($t = 0.4$, $p = 0.69$).

3. A lexicographic model predicts better than a q-compensatory model for more respondents in the full-rank cells than in the consider-then-rank cells (62% vs. 50%, $t = 2.2$, $q = 0.03$).[5]

We obtain a similar pattern of results for hit rates with the exception that hit rates are a coarser measure at the level of the individual respondent (more ties) and require a relative measure.[6]

[4]Using analysis of variance, task is significant for both LBA ($F = 51.1$, $p = 0.00$) and HBRL(4) ($F = 3.7$, $p = 0.05$). Sorting is not significant for either LBA ($F = 2.1$, $p = 0.14$) or HBRL(4) ($F = 0.1$, $p = 0.79$).

[5]This observation is tempered with the realization that the full-rank cells provide more ordered pairs than the consider-then-rank cells (496 vs. 183, on average).

[6]For many respondents the hit-rate prediction of LBA is tied with HBRL(4). Among those that are not tied, significantly more are fit better with LBA in the full-rank cells than in the consider-then-rank cells, $t = 2.3$, $p = 0.02$.

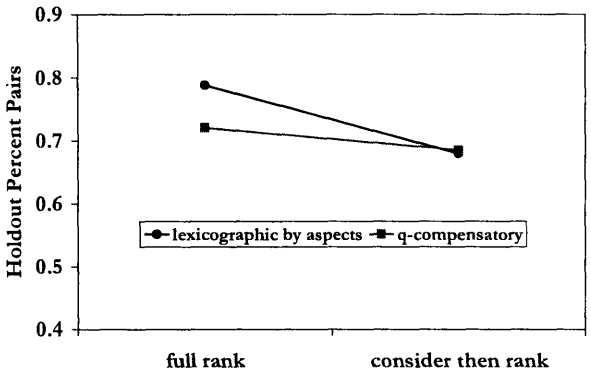

Figure 5-11: Predictive Ability by Experimental Cell, Lexicographic vs. q-Compensatory Processes : Sorting allowed

5.3.6 Constructed Processes : Predictive Ability vs. Effort

Data in the previous section are consistent with a hypothesis that the more-effortful experimental cells (full-rank vs. consider-the-rank) lead to more lexicographic processing. We can also manipulate effort by the number of profiles that the respondent is asked to evaluate. Indeed, behavioral theory suggests that respondents are more likely to use a lexicographic process for choice (rank) if there are more profiles (e.g., Bettman et al [3]; Johnson et al [35]; Lohse and Johnson [43]).

To examine this issue we assigned an additional 86 respondents to a fifth cell in which respondents evaluated fewer profiles (16 vs. 32) using the consider-then-rank task. With this manipulation, we found no significant differences in the relative predictive ability of LBA vs. HBRL(4) between cells ($t = 0.2$, $p = 0.88$ for percent-pairs predicted and $t = 1.0$, $p = 0.31$ for the percent of respondents for whom LBA predicts better). We obtain the same pattern of results with hit rates. Interestingly, the differences in effort are also not significant for 16 vs. 32 profiles when the task is consider-then-rank.[7] Perhaps the number of profiles has less of an effect on consideration than that reported in the literature for choice—an empirical result worth examining in future experiments. Alternatively, the 16 profile task might have already been sufficiently difficult to trigger the use of simplifying heuristics for consideration.

We did not include a cell in which respondents were asked to provide full-ranks

[7]Enjoyment, interest, and accuracy (2.07 vs. 2.04, $t = 0.1$, $p = 0.90$); task time (3.40 vs. 3.75 minutes, $t = 0.5$, $p = 0.64$) for 16 vs. 32 profiles in a consider-then-rank task.

for 16 profiles. However, to gain insight we simulate a 16-profile full-rank cell by randomly choosing one-half of the 32 profiles for estimation. Predictions degrade with half the profiles, but the loss is less than three percentage points (80.8% vs. 77.9%, $t = 4.3$, $p = 0.00$).[8]

The effect of task type seems to have a larger impact than the number of profiles. LBA estimates from the full-rank task predict significantly better than those from the consider-then-rank task (review Figures 5-10 and 5-11). On average (combining sort and no-sort cells), 81% of the holdout pairs are predicted correctly in the full-rank cells compared to 69% in the consider-then-rank cells ($t = 2.6$, $p = 0.01$). On the other hand, the consider-then-rank task took significantly less time to complete in the no-sort cell (8.75 vs. 3.75 minutes).

The three effort comparisons (full-rank vs. consider-then-rank, 16 vs. 32 profiles for consider-then-rank, 16 vs. 32 profiles for full-rank) suggest an interesting managerial tradeoff between predictive ability and task time. With specific loss functions on predictability and task time, such comparisons enable managers to design more efficient market research studies.

5.3.7 Aspects vs. Features

Finally, we address whether respondents process profiles by features or by aspects when they use lexicographic processes. Recall that lexicographic-by-features (LBF) is a restricted form of LBA where respondents rank by features (i.e., all aspects derived from a given feature must appear next to each other in the aspect order). Because LBA nests LBF, LBAs *fit* statistics will be better. However, there is no guarantee that LBAs holdout predictions will be better than those of LBF. If respondents process profiles by features, then LBF may predict as well as LBA, perhaps better if LBA exploits random variations.

Table 5.1 compares LBA to LBF. On average, LBA predicts significantly better

[8]Hit rates are worse by 2.9 percentage points, but the difference is not significant, $t = 1.7$, $p = 0.00$. Because the predicted holdout percentages are based only on the full-rank cells, they differ slightly from those in Table 5.1.

Aspect	ABA or EBA	Affect Consideration*	Top Aspect†
$499	EBA	49.2%	26.1%
Flip	ABA	32.0%	10.4%
Small	ABA	29.4%	10.0%
$299	EBA	19.8%	4.2%
Keyboard	ABA	17.3%	7.5%
$99	ABA	14.5%	4.8%

*Column sums to 300% over all aspects. †Column sums to 100% across all aspects. Most aspects not shown.

Table 5.2: Top Lexicographic Aspects for SmartPhones (for our sample)

on both holdout violations and hit rates. LBA predicts better in all four cells and significantly better in three of the four cells (t's = 1.8, 7.1, 2.4, and 4.5; p's = 0.07, 0.00, 0.02, and 0.00 in Cells 1-4). However, LBF predicts better for about a third of the respondents (35% holdout violations and 34% hit rate, no significant differences between experimental cells).

5.3.8 Managerial Implications

Manufacturers, retailers, or website designers seek to design products, store layouts, or websites that have (or emphasize) those aspects that strongly influence which products customers select for further consideration. They seek to avoid those aspects that customers use to eliminate products. In the parlance of product development, these are the must-have or must-not-have aspects or features (Hauser, Tellis, and Griffin [29]). Both General Motors and Nokia have indicated to us that the identification of must-have aspects is an extremely important goal of their product-development efforts (private communication). Table 5.2 lists the six aspects that were used most often by SmartPhone respondents and indicates whether they were used to retain profiles as in ABA or eliminate profiles as in EBA (second column), the percent of consumers who used that aspect as one of the first three aspects in a lexicographic order (third column), and the percent who used that aspect as the first aspect in a lexicographic order (fourth column).

Table 5.2 has a number of implications. Firstly, for our student sample, there are

clear price segments for almost half the sample high-price is an elimination aspect. Secondly, flip and small are each acceptance aspects for about 30% of the respondents. For this sample, any manufacturer would lose considerable market share if it did not include SmartPhones that were small and flip. The keyboard aspect is interesting. Keyboard is an acceptance aspect for 17.3% of the respondents and an elimination aspect for 7.5% of the respondents (not shown). On this aspect, a manufacturer would be best advised to offer both SmartPhones with keyboards and SmartPhones without keyboards. Finally, brand, service provider, and operating system are not high in the lexicographic ordering.

It is interesting that, in our data, price aspects were often, but not always, elimination aspects, while all other aspects were acceptance aspects. (This is true for aspects not shown in Table 5.2.) We do not know if this generalizes to other categories. Furthermore, although high-price was the top lexicographic aspect in our study, this may be a consequence of the category or our student sample. We do not expect price to be the top lexicographic aspect in all categories nor do we feel that this result affected the basic scientific and methodological findings about lexicographic processing or predictive ability.

5.4 Computers from study by Lenk et al [42]

We were fortunate to obtain a classic conjoint-analysis data set in which respondents evaluated full profiles of computers that varied on thirteen binary features: telephone service hot line, amount of memory, screen size, CPU speed, hard disk size, CD ROM, cache, color, availability, warranty, bundled software, guarantee, and price. Respondents were presented with 16 full profiles and asked to provide a rating on a 10-point likelihood-of-purchase scale. They were then given a holdout task in which they evaluated four additional profiles on the same scale. These data were collected and analyzed by Lenk et al [42], who suggest excellent fit and predictive ability with hierarchical Bayes compensatory models. Based on their analysis and our intuition, we felt that the features in this study were more likely to be compensatory than those

	LBA	HBRL	HBRL ($q = 4$)
Fit (percent pairs)	0.899*	0.906*	0.779
Holdout (percent pairs)	0.790*	0.827**	0.664
Holdout (hit rate)	0.686*	0.692*	0.552

**HBRL significantly better than LBA and HBRL(4). *LBA and HBRL significantly better than HBRL(4). Tests at 0.05 level.

Table 5.3: Comparison of Fit and Prediction for Computer Data (Lenk et al [42])

in the SmartPhone study. However, this is an empirical question.[9]

We first degraded the data from ratings to ranks. For example, if Profile A were rated as a "10" and Profile B were rated as a "1," we retained only that Profile A was preferred to Profile B. Because there were 10 scale points and 16 profiles, there were many ties—an average of 6.6 unique ratings per respondent. Interestingly, even though there were many ties, there were approximately 96 ranked pairs of profiles per respondent—80% of what would be obtained with full ranks. Because the degraded data are partial ranks, we can analyze the data with greedoid methods and compare predictions to HBRL(∞) and HBRL(q).[10]

Table 5.3 reports the fit and prediction results for the computer data. As with the SmartPhone data we address the predictive ability of LBA compared to (1) an unconstrained additive model and (2) a q-compensatory model. On these data, the unconstrained additive model predicts better than LBA, significantly so for holdout pairs. (The difference in hit rates is only one respondent out of 201 respondents.) However, LBA predicts significantly better than the q-compensatory model.

For the computer data, LBA predicts better for 58% of the respondents compared to 25% for HBRL(4); the remainder are tied. We distinguish fewer respondents by hit rate because hit-rate classification is a higher-variance classification: 32% LBA, 20% HBRL(4), and 47% tied.

Interestingly, LBA on the degraded data does as well as *metric* hierarchical Bayes

[9]There are other differences between the data sets that are worth further study. For example, the rating task might induce more compensatory processing than the full-rank or consider-then-rank tasks.

[10]For the Lenk et al data, HBRL predictions are significantly better than those by LINMAP. For holdout pairs, LINMAP predicts 0.734 ($t = 5.3$, $p = 0.00$). For hit rates, LINMAP predicts 0.597 ($t = 2.6$, $p = 0.01$).

on the ratings data (0.687, Lenk et al [42, p. 181]) and better than either OLS (0.637, ibid.) and latent class analysis (0.408, ibid.).[11] In this case, a reduction in effort (ranking vs. rating) may have had little effect on predictive ability. For a further discussion of ranking vs. rating data, see Huber et al [31].

Table 5.3 is consistent with the analysis of metric data by Kohli and Jedidi [39] who found that a different lexicographic model (binary satisficing, LBS) fit almost as well as an unconstrained additive model (0.93 fit pairs for LBS vs. 0.95 for classic LINMAP; no data available on holdouts). The Kohli-Jedidi context is remarkably similar to that of Lenk et al: metric ratings of 16 laptop computers described by memory, brand, CPU speed, hard drive size, and price (in a $3^3 2^2$ fractional design).

Comparing the SmartPhone and computer data, we get surprisingly similar respondent-level comparisons. LBA predicts at least as well as HBRL(4) for 57% of the SmartPhone respondents and 75% of the computer respondents.[12] Kohli and Jedidi [39] did not test a q-compensatory model, but they did find that an unconstrained additive model was not significantly different from LBS for 67% of their respondents. Thus, on all data sets, for more than half of the respondents, noncompensatory models predict holdout data at least as well as q-compensatory models.

We can also compare the predictive ability of LBA to an unconstrained additive model. LBA predicts at least as well as HBRL for 49% of the SmartPhone respondents and 62% of the computer respondents. Thus, even compared to an unconstrained additive model, LBA is promising as a predictive tool.

[11] We compare to the highest hit rate they report that for HB estimated with 12 profiles. For 16 profiles they report a hit rate of 0.670. For other statistics, HB with 16 profiles performs better than with 12 profiles (ibid, p. 181).

[12] The corresponding percentages for hit rates are 71% and 80%.

Chapter 6

Rule-based consideration

In this chapter, we approach consideration set formation from a rule-based perspective, which can be viewed as a nonlinear generalization of lexicography. Given a subject's consideration set, we are interested in finding the best set of rules (consisting of logical expressions) that explain the data while meeting a complexity measure budget constraint. Applying the technique to real data shows that it performs well compared to a compensatory model of consideration set formation, while being the simpler both in terms of complexity and explanation.

6.0.1 Related Work

Roberts and Lattin [55] give an overview of research in consideration. Mehta et al [47] analyze consideration set formation in the context of the cost of information search. They apply the method of simulated moments to estimate the parameters in a random-utility model. Gensch and Soofi [19] use an information-theoretic approach (involving maximizing entropy). Gilbride and Allenby [24] estimate screening rules in Bayesian framework using MCMC methods. Other have also tried to estimate elimination rules as part of the overall choice process (DeSarbo et al. [13], Jedidi and Kohli [32], Jedidi et al. [33], Kim [37], Roberts and Lattin [54], and Swait [65]).

Because consideration set formation can be formulated as a binary classification problem (with considered and not considered classes), existing classification algo-

rithms from the pattern recognition / machine learning fields can be applied. Standard techniques include logistic regression, decision trees, neural networks (NNs), and support vector machines (SVMs). Certain techniques are more discrete in nature and hence more relevant to the current discussion.

Consideration set formation problem bears a strong resemblance to the concept learning task found in machine learning (see Mitchell [48]). In concept learning, which can be viewed as a special case of classification in general, a boolean-valued function is inferred from training examples. Early concept learning algorithms were extremely sensitive to errors in the training data, while later enhancements improved robustness.

Decision trees (e.g., the classification and regression trees (CART) of Breiman et al [8]) can also be viewed as rule-based. In each node of a decision tree, a question is asked and the next decision (e.g., branch left or right) depends on the answer. Once the path through the tree reaches a leaf node, the classification decision corresponding to that leaf is applied. Each path from the root to a leaf corresponding to class k can be expressed as a conjunction of conditions, e.g., "If condition A and condition B and condition C, then assign class k". Furthermore, the collection of all paths from the root to leaf nodes associated with class k can be viewed as a disjunction of conjunctions (of conditions). In this chapter, we will be focusing on this natural form of decision rule.

The main inspiration for our approach comes from a machine learning technique called the Logical Analysis of Data (LAD) developed by Peter Hammer and others (see Boros et al [6]). In LAD, logical rules (boolean expressions) are inferred from training data in order to approximate a boolean function. The main steps are:

1. Generate short positive and negative patterns from the training data.

2. Prune the patterns to form a minimal cover of the training data.

3. Determine weights for the patterns to form an overall/final decision rule.

The overall goal is to discover short logical patterns in the training set that hopefully generalize well to previously unseen observations. When applied to several standard

datasets from the UCI repository, Boros et al [6] report strong performance compared to other top methods. In particular, their single method applied well across many different problems.

Our formulation and algorithms bear a strong resemblance to those in LAD, although adapted and extended for the context of individual choice. Specific differences include:

- Individual level estimation
- Different handling of errors
- Sparseness of data
- Asymmetric accept/reject

These differences require special treatment in our consideration set context.

6.1 Rule-based Model

6.1.1 Notation

As a convention, we will generally use lower case letters for aspects and upper case letters for profiles (which are sets of aspects). We will let C^+ denote the consideration set and C^- represent the set of profiles not considered.

6.1.2 Rule Format

Drawing on decision tree theory, we consider rules that are unions of intersections of variables (aspects), e.g.,

$$(a \wedge b) \vee (c \wedge \neg d \wedge e) \vee (\neg f) \tag{6.1}$$

We call each intersection expression a *pattern* (after LAD), and refer to the whole expression (disjuction of conjunctions) as a *rule* (or consideration rule). We call

patterns derived from (or associated with) profiles in C^+ *positive* patterns and those from profiles in C^- *negative* patterns.

For the rule above, a profile would be said to *match* it if the profile contained aspects a and b, and/or contained aspects c and e but not d, and/or did not contain aspect f. In other words, a rule matches a profile if at least one of its constituent patterns matches that profile.

6.1.3 Decision Rules

There are several possible ways a decision rule can be constructed and used. An *acceptance rule* consists of a set of positive patterns and is applied in the following way: if a profile matches one or more positive patterns, consider it; else do not consider it (the default decision). Conversely, a *rejection rule* consists of a set of negative patterns and has the following interpretation: if a profile matches one or more negative patterns, do not consider it; else consider it (the default decision).

It is an open question as to whether consumers are more likely to use rejection or acceptance rules. Elimination-by-aspects (EBA), one of the early choice heuristic formulations, focused on rejection in the context of choice. The SmartPhone study described in Chapter 5 found that the behavior of respondents in the sample were more consistent with a mixed LBA model than with strict ABA or EBA.

It should be noted that in LAD, both positive and negative patterns are combined (along with weights) into a final decision rule or discriminant function. In essence, each pattern participates in a weighted vote. While this approach is also possible here, it makes the interpretation of the overall rule more difficult and (in our view) less likely to represent how consumers form consideration sets in practice.

6.1.4 Rule Complexity

Note that any training set (consideration set) can be modeled exactly using this rule-based formulation. Specifically, we can construct one pattern per profile where the pattern's positive and negative variables are determined exactly by the presence or

absence of the corresponding aspects in the profile. Such a decision rule is essentially saying, "If the profile is one of the considered profiles already seen, consider it." While this rule achieves perfect performance on the training set, it will likely do poorly on a previously unseen test set. Thus we would like to enforce some regularization on the form or size of the rules in order to improve the chances of good generalization to unseen observations. (See Hastie et al [27] for a discussion of complexity/smoothing parameters and their effect on the bias-variance tradeoff (and test/generalization error).)

As a first pass, we use the total number of aspects across all patterns in a rule as a measure of rule complexity. For example, the rule (6.1) above would have a complexity of 6. An important variation on this theme is counting the total number of *symbols* as opposed to simply aspects, and possibly assigning weights to symbols based on the amount of mental processing power required to carry out the various operations. This variation ties in nicely with the effort-accuracy tradeoff perspective developed in Payne et al [52].

6.1.5 Measure of Goodness

Given a cap on the allowed complexity, how do we decide which of a given number of rules is the best? Standard approaches include symmetric and asymmetric loss functions. In the following section, we use a symmetric loss function such that both types of errors (i.e., incorrectly including a profile and incorrecting excluding a profile) are tallied and counted equally.

6.2 Computational Complexity

Before discussing a method for find the best rule given a complexity budget (i.e., the rule that induces the least number of errors), we show that the problem is NP-hard. First, we show that determining whether or not a perfect rule explaining the consideration data exists is NP-complete.

CONSIDERATION CONSISTENCY	
INSTANCE	: Set of aspects A, set of profiles P, sets $C^+, C^- \subseteq P$, set of (positive) patterns, scalar k
QUESTION	: Is there a rule of complexity $\leq k$ that induces no errors with respect to C^+ and C^-?

Theorem 12 *CONSIDERATION-CONSISTENCY is NP-complete.*

Proof. We reduce CONSIDERATION CONSISTENCY to SET COVER. Suppose we are given SET COVER instance with ground set S containing elements x_1, x_2, ..., x_n, a collection D of subsets of S, D_1, D_2, ..., D_m, and k_{SC}, the upper bound on set cover size. We will construct an instance of CONSIDERATION CONSISTENCY that can be used to solve the SET COVER problem.

Let aspect set A be $\{a_1, a_2, \ldots, a_n\}$, where aspect a_i corresponds to subset D_i. Let profile set P be $\{P_1, P_2, \ldots, P_m, Y_1, Y_2, \ldots, Y_m\}$, where profile P_i is associated with x_i:

$$P_i = \{a_j : x_i \in D_j\},$$

and Y_i are dummy profiles:

$$Y_i = \{\}.$$

Finally, let $C^+ = \{P_i\}$, $C^- = \{Y_i\}$, and $k_{CC} = k_{SC}$.

For example, the SET COVER instance

	D_1	D_2	D_3	D_4
x_1	1	0	1	0
x_2	0	1	0	0
x_3	1	0	0	0
x_4	0	0	1	1

would become

First, suppose there exists a set cover D' with $|D'| = k \leq k_{SC}$. Then a consistent rule exists consisting of k patterns each with a single positive variable corresponding

	a_1	a_2	a_3	a_4
P_1	1	0	1	0
P_2	0	1	0	0
P_3	1	0	0	0
P_4	0	0	1	1
Y_1	0	0	0	0
Y_2	0	0	0	0
Y_3	0	0	0	0
Y_4	0	0	0	0

to the subsets in D'. Because D' is a set cover, all x_i are covered, which implies that all P_i are "matched" and thus cause no errors with respect to C^+. Therefore, if CONSIDERATION CONSISTENCY returns "no", it means that no set cover of size less than or equal to k_{SC} exists.

Conversely, suppose CONSIDERATION CONSISTENCY returns "yes". First note that each pattern must contain only positive variables. (If a pattern contained a negative variable, then all Y_i would be "matched", causing m errors with respect to C^+ and C^-.) Second, note that we can trim each multi-variable pattern down to a single variable (say the first variable). The resulting rule will still be a consistent rule because each shortened pattern is guaranteed to match at least as many profiles P_i as it originally did (and still does not match any profiles Y_i). Finally, because all profiles in C^+ are matched by single-variable rules, it must be the case that the subsets D_i corresponding to the variables in the rules form a set cover in the original problem.

Since CONSIDERATION CONSISTENCY returns a "yes" if and only if the SET COVER instance was a "yes" instance, CONSIDERATION CONSISTENCY is NP-complete. □

We now show that the problem of finding the best consideration rule, i.e., the rule that induces the least number of errors with respect to the data is NP-hard and not in APX.

BEST CONSIDERATION RULE	
INSTANCE	: Set of aspects A, set of profiles P, sets $C^+, C^- \subseteq P$, set of (positive) patterns, scalar k
SOLUTION	: Rule, i.e., a conjuction of patterns, with complexity $\leq k$
MEASURE	: Number of errors with respect to C^+ and C^-

Theorem 13 *BEST-RULE-SET is not in APX (unless P = NP).*

Proof. The result follows directly from Theorem 12. Since determining if achieving zero errors is already hard, it is not possible to approximate the problem of minimizing the number of errors to within a constant factor (because that would mean the decision problem was easy). □

6.3 Estimation Algorithm

6.3.1 Pattern Generation

Given n aspects, there are $2^d\binom{n}{d}$ possible patterns of length d (taking into account both positive and negative orientations–the presence or absence of an aspect). If we restrict ourselves to patterns that actually occur in at least one of the m profiles in C, this bound is reduced to $m\binom{n}{d}$. In either case, exhaustively enumerating all possible patterns of all lengths is neither computationally feasible nor desirable. Instead we generate all patterns that occur in C of length less than or equal to $d = 3$. (For just acceptable patterns, we generate all patterns occurring in C^+.) The justification for this decision is similar to that given for LAD, namely that short expressions are more likely to represent real patterns in the data and have good generalization properties.

6.3.2 Optimization Subject to Complexity Constraint

Given a set of patterns (positive, negative, or both), a complexity "budget", and an objective function (measure of goodness), the task of determining which patterns to

include in a consideration rule can be solved using integer programming.

The following optimization problem minimizes the total number of errors (of both types) induced by a decision rule with complexity less than or equal to k:

$$\begin{aligned}
\text{minimize} \quad & \sum_{i:P_i \in C^+} (1 - z_i) + \sum_{i:P_i \in C^-} z_i \\
\text{subject to} \quad & \sum_{j:T_j \text{ matches } P_k} y_j \geq z_i \qquad \forall i \text{ (profiles)} \\
& \textstyle\sum_j |T_j| y_j \leq k \\
& y_j \in \{0,1\} \qquad \forall j \text{ (patterns)} \\
& z_i \in \{0,1\} \qquad \forall i \text{ (profiles)}
\end{aligned}$$

where T_j is the jth pattern (or template), $|T_j|$ is the complexity (number of aspects) of pattern j, y_j is 1 if pattern j is included in the consideration rule and 0 otherwise, and z_i is 1 if profile i is matched by the consideration rule and 0 otherwise. In the objective function, for profile $i \in C^+$, having $z_i = 1$ (matched) results in no error, while $z_i = 0$ (not matched) results in an error. Similarly, for $i \in C^-$, $z_i = 1$ (matched) results in an error, while $z_i = 0$ (not matched) results in no error.

6.4 SmartPhone Dataset

6.4.1 Benchmarks

As a compensatory benchmark, we use Strict LINMAP [62] augmented with cutoffs. That is, after estimating a partworth vector, all values of a cutoff parameter α are evaluated and the value that minimizes the number of training errors is selected.

We also consider LBA with cutoffs. After an optimal aspect order is found with dynamic programming (or a heuristic), we select the α top aspects (to function as a single acceptance rule) that induce the least number of errors on the training set.

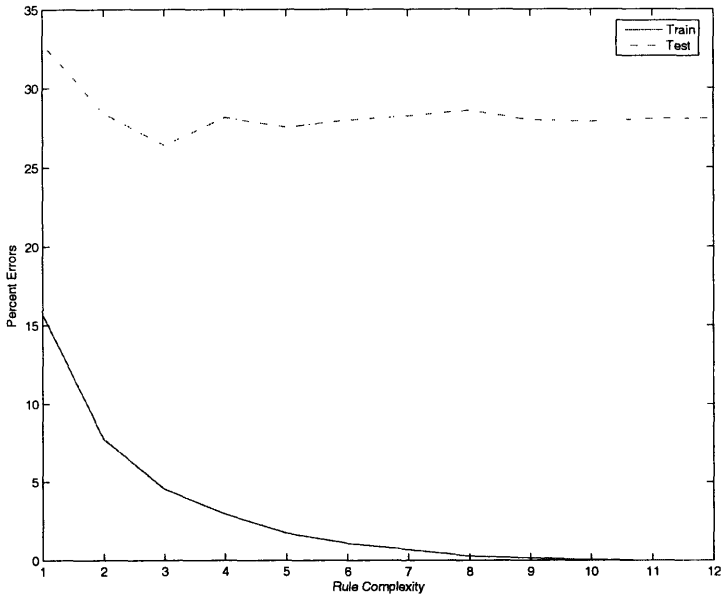

Figure 6-1: Rule-based Training and Test Error Rates

6.4.2 Results

We estimated Strict LINMAP with cutoffs (SLM), LBA with cutoffs (LBA), and rule-based consideration rules for complexities ranging from 1 to 12. The consideration rules were limited to acceptance rules, and the patterns consisted of intersections of up to $d = 3$ aspects. The SmartPhone dataset included over 140 subjects that performed consideration as part of their tasks (and also appeared to fully understand the holdout tasks).

Figure 6-1 shows the training and test (holdout) error rates as rule complexity ranges from 1 to 12. (Note that when a subject's consideration data was already perfectly explained by a rule of size k, that same rule was treated as the best rule for all larger complexities $k + 1$, $k + 2$, etc. The average complexity required to explain all data perfectly was 5.2.).

As can be seen in the figure, the training error rate decreases quickly as the complexity increases. Intuitively, it appears that with each new bit (aspect), the number of errors decreases by a half. Though noisy (due to a smaller number of profiles in the test set vs training set), the test error appears to decreases and then

Method	Train†	Perfect†	Test
SLM	0.998	0.965	0.688
LBA	0.938	0.211	0.730*
Rule2	0.923	0.092	0.716
Rule3	0.954	0.296	0.736*

†All Training values (and Perfect values) significantly different from each other.

*LBA and Rule3 significantly different from SLM and Rule2. All tests at the 0.05 level.

Table 6.1: Comparison of Consideration Set Estimation Methods

level off (perhaps after a slight increase after complexity 3). We might expect (from an understanding of overfitting vs generalization ability) that the test error rate should begin to increase once an optimal complexity level was passed. However, it may be that since the consideration data were perfectly explained after a relatively low complexity anyway (5.2 aspects on average), there was less opportunity for overfitting.

Table 6.1 shows results for SLM, LBA, and rule-based for complexities 2 and 3. (The mean number of aspects in the LBA with cutoffs rules was 3.1.) The columns in the table represent the method, the average fraction of training profiles classified correctly, the fraction of subjects with zero training errors, and the average fraction of test profiles classified correctly.

Even though SLM fit the training data the best, it had the worst test performance. Rule3 and LBA (which had 3.1 aspects on average) predicted holdout consideration the best. It is interesting to note that Rule2 performed as well or better (though not statistically significant) than SLM despite using only 2 bits of informations.

Finally, refer back to Figure 6-1. By complexity 12, the data of every subject is perfectly explained. This is a higher rate than 0.965 achieved by SLM, which indicates that the rule-based formulation was able to capture nonlinearities that the linear compensatory method could not.

6.5 Discussion

Consideration is an important topic in consumer decision making. Choice sets in modern product categories (e.g., digital cameras and cell phones) appear to be growing in

both size and complexity, and most consumers need to use some sort of simplification to wade through them. Thus, analyzing consideration-then-choice is important for understanding decision behavior.

We have proposed a technique for estimating consideration rules built up from short logical expressions. Using integer programming, we were able to choose the best patterns to include in a rule given a complexity constraint (budget). When applied to the SmartPhone dataset, the rule-based approach outperformed a more complex compensatory approach.

There are several possible reasons for the success of the simpler model. Using concepts from machine learning, it could be that the simpler model was more robust in the face of error. Even though the linear compensatory approach could model most consideration behavior perfectly, having so much freedom in choosing the continuous weights might have allowed it to fit the noise or error in the training data. Another possible explanation is that the rule-based approach allows for nonlinear relationships (e.g., interactions between aspects) to be modeled. However, the benefit from this might have been slight for this dataset considering LBA performed on par with rules of complexity 3.

Chapter 7

Conclusion

7.1 Contributions

In this thesis, we have added to both the theory and practice of noncompensatory decision processes. We introduced several new lexicographic models for consideration and choice and a rule-based decision process for consideration. We also proposed a constrained compensatory model that can help classify individuals as truly compensatory or not.

We formulated several noncompensatory inference problems and analyzed their computational complexity. The polynomial nature of some of the problems was shown by applying the theory of greedoid languages (for the first time in the field of marketing science). The harder problems were shown to be not likely approximable to within a constant factor.

Exact greedy algorithms were developed for the easy problems with greedoid structure, while exact dynamic programming algorithms were constructed for the hard problems. In addition, several local search heuristics were analyzed in terms of efficiency and the goodness of the bounds they could supply the dynamic programming algorithm. Ultimately, it was shown that the exact DP method (with enhancements) is robust and scalable. The very fast running times mean that researchers utilizing the method will not be bogged down with computation. Furthermore, future applications where noncompensatory inference must be done on the fly (e.g., web-based

dynamic surveys) seem feasible.

Finally, the methods were tested on several datasets. Not only were the algorithms found to be effective, but several interesting behavioral questions were explored.

7.2 Future Work

Further investigation into noncompensatory inference can proceed along several dimensions. Some of these topics have already begun to be taken up by other researchers.

Clustering Similar to latent class methods in traditional conjoint analysis, it may be beneficial to cluster respondents into several clusters and estimate a single lexicographic strategy for each cluster. One possible way to perform the per-cluster estimation would be to weight each possible pair of profiles according to the observed preferences in the cluster and then solve the MIN WEIGHTED ERRORS problem.

Nearest Neighbor Shrinkage In Hierarchical Bayes style approaches, population-level information is used to improve individual-level estimation. A possible extension to noncompensatory inference (similar to the previous point on clustering) would be to adjust weights for profile pairs perferences to be a combination of the individual's observed preferences and those of other individuals that are "close" by according to some metric.

Hybrid Decision Processes Perhaps the most interesting extension would be to consider individuals with mixed or hybrid decision strategies, i.e., sometimes noncompensatory and sometimes compensatory. It seems possible to estimate a hybrid model that allows only a certain number of aspects to be lexicographic with integer programming.

Appendix A

Experimental Designs

A 32×20 design was generated in MATLAB with the following commands:

```
nr = 32;
nc = 20;
X = cordexch(nc, nr);
```

The design (with 1 and -1 recoded to 1 and 0) is given in Table A. The D-efficiency and A-efficiency of the design (when coded in were measured with the following commands:

```
deff = 1 / (nr * det(inv(X'*X))^(1/nc))
aeff = 1 / (nr * trace(inv(X'*X)) / nc)
```

yielding

```
deff = 0.97690
aeff = 0.95347
```

In a similar way, the 64×40 design shown in Tables A and A was generated. This design had

```
deff = 0.97414
aeff = 0.94849
```

0 0 0 1 1 0 0 0 1 0 0 0 0 1 0 0 1 0 0 0
0 1 0 0 1 0 0 0 0 0 1 1 0 1 0 0 0 1 0 1
1 1 0 1 1 1 1 0 1 0 1 0 1 1 0 1 0 0 0 1
1 0 0 1 0 1 0 0 1 0 0 0 0 0 1 0 0 1 0 0
0 1 0 1 0 1 1 0 0 0 1 1 1 1 1 0 1 0 1 0
1 1 1 1 0 0 0 1 0 1 1 1 0 1 0 1 0 1 0 0
0 0 0 1 0 0 1 0 1 1 1 1 0 0 1 1 0 1 1 1
1 1 0 0 0 0 1 0 1 1 1 0 1 0 0 0 0 0 1 0
1 1 1 1 0 0 1 1 1 1 1 0 0 1 1 0 1 0 0 0
0 1 0 0 1 1 1 0 0 1 0 1 0 0 1 1 1 0 0 0
1 1 0 1 0 0 0 0 0 1 0 0 1 0 1 1 1 1 0 1
0 1 1 1 1 0 0 1 1 0 1 1 0 0 1 0 1 0 1 1
0 1 1 1 1 1 1 0 1 1 0 1 1 0 1 0 0 1 0 0
0 1 1 0 0 0 1 1 1 0 0 1 1 0 0 0 1 1 0 1
1 1 1 0 0 1 0 0 1 1 0 1 0 1 1 1 0 0 1 1
1 1 0 1 1 1 1 1 0 1 0 0 0 1 0 0 1 1 1 1
0 0 0 0 1 1 0 1 1 1 1 0 1 1 1 1 1 1 0 0
0 1 1 0 0 0 0 0 0 0 0 0 1 1 0 1 0 0 1 0
1 0 1 1 1 0 0 1 0 1 0 1 1 0 0 0 0 0 0 0
1 0 0 0 1 0 1 1 0 0 0 1 1 1 1 0 0 1 1 1
0 1 1 1 1 1 1 1 1 0 0 0 0 0 0 1 0 1 1 0
0 0 0 0 0 1 0 1 0 1 1 1 0 0 0 0 0 0 1 0
0 0 1 1 0 1 1 1 0 0 1 0 1 1 1 1 0 0 0 1
1 0 1 1 1 1 1 0 0 0 1 1 0 0 0 1 1 0 0 1
0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 1 1 0 0 1
1 1 0 1 0 1 0 1 1 0 1 1 1 0 0 1 1 1 1 0
1 0 1 0 1 1 0 0 1 1 1 0 1 0 0 0 1 0 1 1
1 0 1 0 1 0 1 0 0 0 1 0 0 0 1 1 1 1 1 0
1 1 0 0 0 1 0 1 0 0 0 0 0 0 1 0 0 0 0 1
1 0 0 1 1 0 0 1 1 0 0 1 1 1 1 1 1 0 1 0
0 0 1 1 0 1 0 0 0 1 0 0 1 1 0 0 1 1 1 1
1 0 1 0 0 1 1 0 1 0 0 1 0 1 0 0 1 1 0 0

Table A.1: 32×20 experimental design

0 1 1 0 1 0 1 0 1 1 0 1 1 0 1 1 1 1 0 0 1 0 1 0 1 1 0 1 1 0 1 1 1 0 0 1 1 0 1 1
1 0 1 0 1 1 1 0 1 1 1 0 0 0 0 1 0 1 0 1 1 0 1 1 0 1 1 0 1 0 1 0 1 0 1 0 0 0 0 0
0 0 1 0 1 1 1 0 0 1 0 1 1 0 1 0 1 0 1 1 1 1 1 0 0 1 0 1 0 0 1 1 0 0 1 1 0 0 0 1 1
1 0 0 0 0 1 1 0 0 1 1 1 0 0 0 0 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 1 0 0 1 0 1 1 0 1
1 1 1 0 0 0 1 0 0 0 0 0 0 0 1 0 1 0 1 1 1 0 1 1 0 0 0 1 0 0 0 1 0 1 0 0 1 1 1 1 0
0 1 1 1 0 0 1 1 1 0 1 0 0 1 1 1 1 0 0 1 1 1 0 1 1 0 1 0 0 0 0 0 0 0 0 1 1 0 0 0
0 1 0 0 0 1 0 0 0 1 0 1 1 1 0 0 0 0 1 0 1 0 1 0 1 1 1 1 1 0 0 0 1 0 0 1 0 0 1 0
0 1 1 1 0 1 0 1 1 0 1 0 0 0 1 0 1 0 1 1 1 0 1 0 0 1 1 1 1 0 1 1 1 0 1 1 1 1 1 1
1 0 1 1 0 0 1 1 0 1 0 1 0 0 0 1 1 0 0 1 1 1 1 1 1 0 1 1 1 0 0 0 1 0 1 0 1 0 1 1 1
0 0 1 0 1 1 1 0 1 0 1 1 1 1 0 0 1 0 1 0 0 1 1 0 0 0 0 0 1 1 0 1 1 1 0 1 1 1 0 1
1 1 1 1 1 0 1 0 1 0 0 1 0 1 0 0 0 1 1 0 1 1 0 1 0 0 1 1 1 1 0 1 0 0 0 0 0 0 0 1 1
0 0 0 1 1 0 0 1 0 1 1 1 0 0 0 1 0 0 1 0 0 0 1 0 0 0 0 0 0 0 1 1 0 0 1 0 1 0 0 0 1
1 0 1 1 1 1 0 1 0 0 0 0 1 0 0 1 1 0 0 0 0 0 0 1 1 1 1 0 0 1 1 1 1 0 0 1 0 0 1 1 0
0 0 0 1 1 0 0 1 1 1 0 0 1 0 1 1 0 1 1 1 0 1 1 0 1 1 1 1 0 1 0 1 1 0 0 0 0 1 0 1
1 1 0 1 1 1 1 1 1 1 0 0 0 0 1 0 0 1 1 0 1 0 1 0 1 0 0 0 0 0 1 0 0 1 0 1 1 1 1 1
1 1 0 1 1 1 1 0 0 0 0 1 1 1 1 1 1 0 1 0 1 1 0 1 0 1 1 0 0 0 1 1 1 0 1 0 1 0 1 1
0 0 0 0 1 1 1 1 1 0 1 1 0 1 1 1 0 0 0 1 1 1 1 1 1 1 0 1 1 1 0 1 1 1 1 1 1 1 0 0 0 1
1 1 0 0 1 0 1 1 0 1 1 0 1 1 0 0 1 1 0 0 0 1 0 0 1 1 1 0 1 0 0 1 1 1 0 0 0 1 0 1
0 0 1 1 1 0 1 1 0 1 0 1 1 1 1 0 0 1 0 1 0 0 0 0 0 1 1 0 1 0 0 0 0 0 1 0 1 1 0 0
0 1 0 1 0 1 1 0 1 0 1 1 0 0 1 0 0 0 0 1 0 1 1 0 0 1 0 1 0 1 1 0 0 0 0 0 0 0 1 1 0
1 0 0 1 1 0 0 0 0 0 1 1 1 1 0 0 1 0 0 1 1 0 1 1 1 0 1 1 1 1 0 0 0 0 1 1 1 1 1 1
0 1 0 0 1 0 0 1 1 1 0 1 0 0 0 1 1 0 0 0 1 1 0 0 0 0 0 1 0 0 0 0 0 0 1 1 0 1 0 0
0 0 1 0 1 0 1 1 1 1 1 0 0 1 1 1 0 0 1 1 0 0 0 1 1 1 0 1 1 1 1 1 0 0 0 0 1 1 1 1
1 1 1 0 0 0 0 1 1 0 1 1 1 0 1 1 1 1 1 0 0 0 1 0 1 0 1 0 0 0 0 0 0 1 1 0 1 0 1 1
0 0 1 1 0 0 0 0 0 0 1 1 1 1 0 1 0 0 1 0 1 1 1 0 1 1 0 0 1 0 1 0 1 0 1 0 0 1 0 1
1 1 1 0 1 0 0 0 0 0 1 1 0 0 1 0 0 1 0 0 0 1 0 1 0 1 1 0 0 1 1 0 1 0 0 1 1 1 0 1
0 0 0 0 0 0 0 0 1 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 1 1 1 0 0 0 1 1 0 1 1 1 1 0 1 1
1 0 0 0 0 0 1 1 0 1 1 0 1 1 1 0 1 0 1 0 1 0 1 1 0 0 1 1 0 0 1 0 1 1 0 0 0 1 1 0
1 1 1 0 1 1 0 0 0 0 1 1 1 0 1 1 0 0 0 0 1 0 0 0 1 1 1 1 0 1 0 1 0 1 0 1 0 1 1 0
1 0 1 1 0 1 0 0 1 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 1 0 0 0 1 0 0 1
0 1 0 0 1 0 1 0 1 0 1 1 0 0 0 0 1 0 1 1 0 0 0 1 1 1 0 0 0 0 0 1 1 1 1 0 0 0 1 0
0 1 1 1 1 1 0 0 0 1 1 0 0 1 0 1 1 1 0 0 1 1 0 0 1 0 0 1 0 1 1 0 1 1 1 0 1 1 1 0

Table A.2: First half of 64×40 experimental design

1 0 0 0 1 0 0 0 0 1 1 0 0 1 1 1 1 0 0 1 1 0 0 0 0 1 0 0 1 1 0 0 1 1 0 1 1 0 1 1
0 1 1 0 1 1 0 1 0 0 0 0 1 1 0 0 0 1 1 1 1 1 1 1 0 1 0 1 0 0 0 1 0 1 1 1 1 1 0 0
1 1 1 1 1 0 0 0 1 1 1 1 1 1 1 0 0 1 0 1 0 1 1 1 0 1 0 0 1 0 1 0 0 1 1 1 0 0 1 0
0 0 0 0 0 0 1 1 0 0 1 0 1 0 0 0 0 1 0 0 1 1 0 0 0 1 0 1 1 0 1 1 0 1 1 1 1 0 1 0
1 0 1 0 1 0 0 1 1 0 0 1 0 0 0 0 1 1 0 1 1 1 1 0 1 1 1 1 0 1 1 0 1 1 0 0 1 0 0 0
0 1 0 1 0 1 1 1 0 0 0 0 1 0 1 0 1 0 0 0 0 1 1 1 1 1 1 0 1 1 0 0 1 1 0 1 1 0 0 0
1 0 0 1 1 1 0 1 1 0 0 1 0 1 1 0 0 0 0 0 1 0 0 1 1 0 0 0 0 0 1 1 1 0 0 0 0 1 0 0
0 0 1 0 0 1 0 0 1 1 1 0 0 1 1 0 1 1 0 0 1 0 1 0 1 0 1 0 0 1 0 1 0 1 0 0 0 1 0 1 0 0
1 1 1 0 0 1 0 1 0 1 0 1 0 0 1 0 0 0 0 1 1 0 0 1 0 0 0 1 1 1 0 0 1 1 1 0 1 1 0 1
1 1 0 0 0 0 0 0 1 1 0 0 0 0 1 1 1 0 1 0 1 1 0 0 0 1 1 1 1 1 1 1 0 0 1 0 1 0 0 0
0 0 0 0 1 0 0 0 1 0 0 0 1 0 1 1 0 0 1 1 1 1 0 1 1 0 0 0 1 0 0 0 1 1 0 0 1 1 1 0
1 0 0 0 0 0 0 0 1 0 0 1 0 1 1 1 1 1 1 0 0 0 1 1 1 1 0 1 1 1 0 0 1 1 1 1 0 1 0 0
1 0 1 1 0 1 1 1 1 1 1 1 0 1 0 1 1 0 1 0 0 1 0 0 1 1 0 1 1 0 0 0 0 1 0 0 1 0 1 0
0 1 0 0 0 1 0 1 0 0 0 1 0 1 0 1 1 1 0 1 0 0 0 0 0 0 1 0 1 0 1 1 1 1 0 0 0 1 1 1
1 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 1 0 1 1 1 0 0 0 0 0 0 0 1 0 0 1 0
0 1 0 0 0 0 1 1 0 0 1 0 1 0 0 1 0 1 0 1 1 0 1 1 1 0 0 1 0 1 0 0 1 0 0 0 0 0 0 1 1
0 0 0 0 1 1 0 1 0 0 1 0 0 1 1 0 1 1 1 0 0 1 1 1 0 1 1 1 0 1 0 0 0 0 1 0 0 0 1 1
0 0 1 1 0 1 1 0 1 1 0 0 1 0 1 1 1 1 0 0 0 1 0 1 0 0 0 1 0 0 0 0 1 1 1 1 0 1 1 1
1 0 0 0 1 1 1 1 1 1 0 1 1 1 0 1 0 0 0 1 0 1 1 0 0 0 0 0 0 1 0 1 1 0 1 1 1 0 1 0
1 1 0 0 0 1 1 0 1 0 0 0 1 1 0 1 0 0 0 1 0 0 0 0 1 1 1 1 0 1 1 0 0 1 1 0 1 1 0 1
0 1 0 1 0 0 0 0 1 1 0 0 1 1 0 0 1 1 0 1 1 1 1 1 1 0 0 0 0 1 1 1 1 0 1 0 0 0 0 0 1
0 0 0 1 0 0 1 0 0 0 1 1 0 0 1 1 0 1 0 0 1 0 1 0 0 0 1 0 1 1 0 1 0 1 1 0 1 1 1 0
1 1 0 1 1 1 1 1 1 0 1 0 1 0 0 1 0 0 1 1 1 0 0 0 0 1 1 0 1 1 0 0 0 1 1 1 0 1 0 1
0 1 1 1 1 0 1 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 1 1 1 0 1 1 1 1 0 1 1 1 1 0 1 0 0 0
1 0 1 1 0 0 1 1 0 0 0 0 0 1 1 0 0 1 1 1 1 1 0 0 1 1 0 0 0 1 0 1 1 1 1 1 0 0 1 1
1 0 0 1 0 1 0 0 0 1 1 1 1 0 1 1 0 1 1 1 0 1 0 1 1 0 1 1 1 0 1 1 0 1 0 1 1 0 0 0
0 0 1 0 0 1 1 1 0 0 0 1 1 0 0 1 1 1 1 0 1 0 0 1 0 1 0 0 1 1 1 0 0 0 0 1 0 0 0 1
1 0 1 0 1 0 1 0 0 0 0 0 1 1 1 1 1 0 1 1 0 0 1 0 0 0 0 1 0 0 1 1 0 1 0 0 0 0 0 0
1 0 1 0 0 0 0 1 1 1 1 0 1 0 0 0 0 0 1 0 0 1 0 1 1 0 1 0 1 1 1 1 1 0 1 1 0 1 1 0
1 1 0 1 0 0 1 1 0 1 1 1 1 1 1 0 1 1 1 1 0 0 0 0 1 0 0 1 0 1 1 1 1 0 1 1 0 0 0 0
1 1 1 0 0 1 0 1 0 1 0 0 0 1 1 1 0 0 0 0 0 1 1 0 1 0 0 0 1 0 0 1 0 0 1 0 0 0 1 1
0 1 1 1 0 1 0 1 1 1 1 1 1 1 0 1 0 1 1 1 1 0 0 1 0 1 1 0 0 1 0 1 1 1 0 0 1 0 1 0

Table A.3: Second half of 64×40 experimental design

Bibliography

[1] G. Ausiello, P. Crescenzi, G. Gambosi, V. Kann, A. Marchetti-Spaccamela, and M. Protasi. *Complexity and Approximation: combinatorial optimization problems and their approximability properties.* Springer, New York, 1999.

[2] Moshe Ben-Akiva and Steven R. Lerman. *Discrete Choice Analysis.* MIT Press, Cambridge, MA, 1985.

[3] James R. Bettman, Mary Frances Luce, and John W. Payne. Constructive consumer choice processes. *Journal of Consumer Research*, 25(3):187–217, December 1998.

[4] James R. Bettman and L. W. Park. Effects of prior knowledge and experience and phase of the choice process on consumer decision processes: a protocol analysis. *Journal of Consumer Research*, 7:234–248, 1980.

[5] Anders Bjorner and Gunter M. Ziegler. Introduction to greedoids. In N. White, editor, *Matroid applications*, pages 284–357. Cambridge University Press, London, 1992.

[6] Endre Boros, Peter L. Hammer, Toshihide Ibaraki, Alexander Kogan, Eddy Mayoraz, and Ilya Muchnik. An implementation of logical analysis of data. *IEEE Transactions on Knowledge and Data Engineering*, 12(2):292–306, March/April 2000.

[7] Ernest Andrew Boyd. *Optimization problems on greedoids.* PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, 1988.

[8] L. Breiman, J. H. Friedman, R. A. Olshen, and P. J. Stone. *Classification and Regression Trees.* Wadsworth International Group, Belmont, CA, 1984.

[9] Arndt Bröder. Assessing the empirical validity of the "take the best" heuristic as a model of human probabilistic inference. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 26(5):1332–1346, 2000.

[10] Valerie M. Chase, Ralph Hertwig, and Gerd Gigerenzer. Visions of rationality. *Trends in Cognitive Sciences*, 2(6):206–214, June 1998.

[11] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms.* MIT Press, 2nd edition, 2001.

[12] Jean Czerlinski, Gerd Gigerenzer, and Daniel G. Goldstein. How good are simple heuristics? In Gerd Gigerenzer, Peter M. Todd, and the ABC Research Group, *Simple Heuristics That Make Us Smart*, pages 97-118. Oxford University Press, New York, 1999.

[13] Wayne S. DeSarbo, Donald R. Lehmann, Greg Carpenter, and I. Sinha. A stochastic multidimensional unfolding approach for representing phased decision outcomes. *Psychometrika*, 61:485–508, September 1996.

[14] Hillel J. Einhorn. The use of nonlinear, noncompensatory models in decision making. *Psychological Bulletin*, 73(3):221–230, 1970.

[15] Hillel J. Einhorn and Robin M. Hogarth. Behavioral decision theory: processes of judgment and choice. *Annual Review of Psychology*, 32:52–88, 1981.

[16] U. Feige. A threshold of ln n for approximating set cover. *Journal of the ACM*, 45:634–652, 1998.

[17] Michael R. Garey and David S. Johnson. *Computers and Intractability: a guide to the theory of NP-completeness.* W. H. Freeman, San Francisco, 1979.

[18] Dennis H. Gensch. A two-stage disaggregate attribute choice model. *Marketing Science*, 6:223–231, Summer 1987.

[19] Dennis H. Gensch and Ehsan S. Soofi. Information-theoretic estimation of individual consideration sets. *International Journal of Research in Marketing*, 12:25–38, May 1995.

[20] Gerd Gigerenzer and Daniel G. Goldstein. Reasoning the fast and frugal way: Models of bounded rationality. *Psychological Review*, 1003(4):650–669, 1996.

[21] Gerd Gigerenzer and Reinhard Selten. Rethinking rationality. In Gerd Gigerenzer and Reinhard Selten, editors, *Bounded rationality: the adaptive toolbox*, pages 1–12. The MIT Press, Cambridge, MA, 2001.

[22] Gerd Gigerenzer and Peter M. Todd. Fast and frugal heuristics: the adaptive toolbox. In Gerd Gigerenzer, Peter M. Todd, and the ABC Research Group, *Simple Heuristics That Make Us Smart*, pages 1-12. Oxford University Press, New York, 1999.

[23] Gerd Gigerenzer, Peter M. Todd, and ABC Research Group. *Simple heuristics that make us smart.* Oxford University Press, New York, 1999.

[24] Timothy Gilbride and Greg M. Allenby. A choice model with conjunctive, disjunctive, and compensatory screening rules. *Marketing Science*, 23(3):391–406, Summer 2004.

[25] Paul E. Green, Abba M. Krieger, and Pradeep Bansal. Completely unacceptable levels in conjoint analysis: a cautionary note. *Journal of Marketing Research*, 25:293–300, August 1988.

[26] Paul E. Green, Abba M. Krieger, and Yoram (Jerry) Wind. Thirty years of conjoint analysis: Reflections and prospects. *Interfaces*, 31(3 - Supplement):56–73, May-June 2001.

[27] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The elements of statistical learning.* Springer-Verlag, New York, 2001.

[28] John R. Hauser. Testing the accuracy, usefulness and significance of probabilistic models: an information theoretic approach. *Operations Research*, 26(3):406–421, May-June 1978.

[29] John R. Hauser, Gerald Tellis, and Abbie Griffin. Research on innovation: a review and agenda for marketing science. forthcoming, *Marketing Science*, 2005.

[30] John R. Hauser and Birger Wernerfelt. An evaluation cost model of consideration sets. *Journal of Consumer Research*, 16:393–408, March 1990.

[31] Joel Huber, Dan Ariely, and Greg Fischer. Expressing preferences in a principal-agent task: a comparison of choice, rating and matching. *Organizational Behavior and Human Decision Processes*, 87(1):66–90, January 2002.

[32] Kamel Jedidi and Rajeev Kohli. Probabilistic subset-conjunctive models for heterogeneous consumers, November 2004. Working Paper, Graduate School of Business, Columbia University.

[33] Kamel Jedidi, Rajeev Kohli, and Wayne S. DeSarbo. Consideration sets in conjoint analysis. *Journal of Marketing Research*, 33:364–372, August 1996.

[34] Eric J. Johnson and Robert J. Meyer. Compensatory choice models of non-compensatory processes: The effect of varying context. *Journal of Consumer Research*, 11(1):528–541, June 1984.

[35] Eric J. Johnson, Robert J. Meyer, and Sanjoy Ghose. When choice models fail: compensatory models in negatively correlated environments. *Journal of Marketing Research*, 26:255–290, August 1989.

[36] Richard Johnson. Comment on 'adaptive conjoint analysis: some caveats and suggestions'. *Journal of Marketing Research*, 28:223–225, May 1991.

[37] Jin Gyo Kim. Dynamic heterogeneous choice heuristics: A bayesian hidden markov mixture model approach. Working Paper, MIT Sloan School of Management, Cambridge, MA, 2004.

[38] Noreen M. Klein. Assessing unacceptable attribute levels in conjoint analysis. *Advances in Consumer Research*, XIV:154–158, 1988.

[39] Rajeev Kohli and Kamel Jedidi. Representation and inference of lexicographic preference models and their variants. Working Paper, Columbia University, New York, NY, December 2004.

[40] Bernhard Korte and Laszlo Lovasz. Basis graphs of greedoid and two-connectivity. *Mathematical Programming Study*, 24:158–165, 1985.

[41] Bernhard Korte, Laszlo Lovasz, and Rainer Schrader. *Greedoids*, volume 4 of *Algorithms and Combinatorics*. Springer-Verlag, Berlin, 1991.

[42] Peter J. Lenk, Wayne S. DeSarbo, Paul E. Green, and Martin R. Young. Hierarchical bayes conjoint analysis: recovery of partworth heterogeneity from reduce experimental designs. *Marketing Science*, 15(2):173–191, 1996.

[43] Gerald J. Lohse and Eric J. Johnson. A comparison of two process tracing methods for choice tasks. *Organizational Behavior and Human Decision Processes*, 68(1):28–43, October 1996.

[44] Mary Frances Luce, John W. Payne, and James R. Bettman. Emotional trade-off difficulty and choice. *Journal of Marketing Research*, 36:143–159, 1999.

[45] Laura Martignon and Ulrich Hoffrage. Fast, frugal, and fit: Simple heuristics for paired comparisons. *Theory and Decision*, 52:29–71, 2002.

[46] Laura Martignon and Michael Schmitt. Simplicity and robustness of fast and frugal heuristics. *Minds and Machines*, 9:565–593, 1999.

[47] Nitin Mehta, Surendra Rajiv, and Kannan Srinivasan. Price uncertainty and consumer search: a structural model of consideration set formation. *Marketing Science*, 22(1):58–84, Winter 2003.

[48] Tom Mitchell. *Machine Learning*. McGraw-Hill, New York, 1997.

[49] H. Montgomery and O. Svenson. On decision rules and information processing strategies for choices among multiattribute alternatives. *Scandinavian Journal of Psychology*, 17:283–291, 1976.

[50] Jose Nino-Mora. On certain greedoid polyhedra, partially indexable scheduling problems, and extended restless bandit allocation indices. Economics Working Paper 456, Department of Economics and Buisness, Universitat Pompeu Fabra, 2000.

[51] John W. Payne. Task complexity and contingent processing in decision making: an information search. *Organizational Behavior and Human Performance*, 16:366–387, 1976.

[52] John W. Payne, James R. Bettman, and Eric J. Johnson. *The Adaptive Decision Maker*. Cambridge University Press, Cambridge, UK, 1993.

[53] Robert S. Pindyck and Daniel L. Rubinfeld. *Microeconomics*. Pearson Prentice Hall, Upper Saddle River, NJ, 6 edition, 2005.

[54] John H. Roberts and James M. Lattin. Development and testing of a model of consideration set composition. *Journal of Marketing Research*, 28:429–440, November 1991.

[55] John H. Roberts and James M. Lattin. Consideration: review of research and prospects for future insights. *Journal of Marketing Research*, 34(3):406–410, August 1997.

[56] Peter E. Rossi and Greg M. Allenby. Bayesian statistics and marketing. *Marketing Science*, 22(3):304–328, Summer 2003.

[57] Michael Schmitt and Laura Martignon. Complexity of lexicographic strategies on binary cues. unpublished, 1999.

[58] Steven M. Shugan. The cost of thinking. *Journal of Consumer Research*, 7(2):99–111, September 1980.

[59] Herbert A. Simon. Rational choice and the structure of environments. *Psychology Review*, 63:129–138, 1956.

[60] Herbert A. Simon. *Models of man.* Wiley, New York, 1957.

[61] Herbert A. Simon. Invariants of human behavior. *Annual Review of Psychology*, 41:1–19, 1990.

[62] V. Srinivasan. A strict paired comparison linear programming approach to nonmetric conjoint analysis. In Jay E. Aronson and Stanley Zionts, editors, *Operations Research: Methods, Models, and Applications.* Quorum Books, Westport, CT, 1998.

[63] V. Srinivasan and Allan Shocker. Linear programming techniques for multidimensional analysis of preferences. *Psychometrika*, 38(3):337–369, September 1973.

[64] V. Srinivasan and Gordon A. Wyner. Casemap: computer-assisted self-explication of multiattributed preferences. In W. Henry, M. Menasco, and K. Takada, editors, *Handbook on new product development and testing*, pages 91–112. D. C. Heath, Lexington, MA, 1988.

[65] Joffre Swait. A noncompensatory choice model incorporating cutoffs. *Transportation Research B*, 35:903–928, 2001.

[66] Olivier Toubia, Duncan I. Simester, John R. Hauser, and Ely Dahan. Fast polyhedral adaptive conjoint estimation. *Marketing Science*, 22(3):272–303, Summer 2003.

[67] Amos Tversky. Elimination by aspects: A theory of choice. *Psychological Review*, 79:281–299, 1972.

[68] Glen L. Urban and John R. Hauser. 'listening-in' to find and explore new combinations of customer needs. *Journal of Marketing*, 68:72–87, April 2004.

[69] Jerry Wind, Paul E. Green, Douglas Shifflet, and Marsha Scarbrough. Courtyard by marriott: designing a hotel facility with consumer-based marketing models. *Interfaces*, 19(1):25–47, 1989.

[70] David Wolpert and William G. Macready. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82, April 1997.