Supplemental Appendices for

Greedoid-Based Non-compensatory Inference

Michael Yee

Ely Dahan

John R. Hauser

James Orlin

April 2006

These appendices are provided as supplements to "Greedoid-based Non-compensatory Inference." The appendices are available from the authors and on the *Marketing Science* website.

1. Illustrative example of a simple application of the greedoid dynamic program.

2. Predictive ability of q-compensatory models as q increases

3. Monte Carlo Experiments

Michael Yee is a graduate student at the Operations Research Center, E40-149, Massachusetts Institute of Technology, 77 Massachusetts Avenue, Cambridge, MA 02139, (617) 253-6185, myee@mit.edu.

Ely Dahan is an Assistant Professor of Marketing at the Anderson School, University of California at Los Angeles, 110 Westwood Plaza, B-514, Los Angeles, CA 90095, (310) 206-4170, fax (310) 206-7422, edahan@ucla.edu.

John R. Hauser is the Kirin Professor of Marketing and Head, Management Science Area, MIT Sloan School of Management, Massachusetts Institute of Technology, E56-314, 38 Memorial Drive, Cambridge, MA 02142, (617) 253-2929, fax (617) 253-7597, jhauser@mit.edu.

James Orlin is the Edward Pennell Brooks Professor of Operations Research and Co-director, MIT Operations Research Center, Massachusetts Institute of Technology, E40-147, 77 Massachusetts Avenue, Cambridge, MA 02139, (617) 253-6606, jorlin@mit.edu.

Supplemental Appendix 1: Illustrative Example of Greedoid Dynamic Program

In the text we illustrate elimination-by-aspects (EBA), acceptance-by-aspects (ABA) and lexicographic-by-aspects (LBA) based on SmartPhone profiles. We provide here a second more-visual example based on playing cards.

Figure SA1
Playing-Card Examples of Lexicographic Heuristic Processes

Simplifying Heuristic	TLA (*Three-Letter Acronym*)	Ranking Rule	First Choice	by 2nd Choice	3rd Choice	Totally Diverge by 4th Choice	5th Choice	6th Choice	7th Choice	Last Choice
Lexicographic By Features	LBF	(♠ > ♥ > ♦ > ♣), (A > J)	A♠	J♠	A♥	J♥	A♦	J♦	A♣	J♣
Acceptance By Aspects	ABA	♠, A, ♥, ♦	A♠	J♠	A♥	A♦	A♣	J♥	J♦	J♣
Elimination By Aspects	EBA	⊗♣, ⊗J, ⊗♦, ⊗♥	A♠	A♥	A♦	J♠	J♥	J♦	A♣	J♣
Lexicographic By Aspects	LBA	⊗♣, ♠, A, ♥	A♠	J♠	A♥	A♦	J♥	J♦	A♣	J♣

Suppose that we represent playing cards by two letters where S = spades, H = hearts, D = diamonds, C = clubs, A = ace and J = jack. Then, if we look at the acceptance-by-aspects row of Figure SA1, the playing cards are ordered as follows.

$$AS \succ JS \succ AH \succ AD \succ AC \succ JH \succ JD \succ JC$$

The greedoid dynamic program algorithm (Algorithm 2) generates the following table. For each subset of aspects, we compute the minimum number of errors (see examples in the following section), and record the set of aspects that can occur in the last position to achieve this least error. We begin with all singleton subsets of aspects, then all doubletons, etc. Because some aspects are linked in features, {A, J} or {S, H, D, C}, there is a redundancy that we would eliminate in an efficient code. For ease of exposition, we retain all subsets in the table below.

Subset s	Min Errors *J*(s)	Best Last Aspect
{H}	6	H
{D}	8	D
{C}	10	C
{S}	0	S
{A}	3	A
{J}	13	J
{H, D}	11	D
{H, C}	13	C
{H, S}	2	H
{H, A}	5	H

{H, J}	14	J
{D, C}	15	C
{D, S}	4	D
{D, A}	7	D
{D, J}	16	J
{C, S}	6	C
{C, A}	9	C
{C, J}	18	J
{S, A}	0	A
{S, J}	10	J
{A, J}	3	J
{H, D, C}	15	C
{H, D, S}	3	D
{H, D, A}	7	D
{H, D, J}	16	D or J
{H, C, S}	5	C
{H, C, A}	9	C
{H, C, J}	18	C or J
{H, S, A}	0	H
{H, S, J}	8	J
{H, A, J}	5	H or J
{D, C, S}	7	C
{D, C, A}	11	C
{D, C, J}	20	C or J
{D, S, A}	2	D
{D, S, J}	10	J
{D, A, J}	7	D or J
{C, S, A}	4	C
{C, S, J}	12	J
{C, A, J}	9	C or J
{S, A, J}	0	J
{H, D, C, S}	3	C
{H, D, C, A}	9	C
{H, D, C, J}	18	C
{H, D, S, A}	0	D
{H, D, S, J}	7	J
{H, D, A, J}	7	D or J
{H, C, S, A}	2	C
{H, C, S, J}	9	J
{H, C, A, J}	9	C or J
{H, S, A, J}	0	H or J
{D, C, S, A}	4	C
{D, C, S, J}	11	J
{D, C, A, J}	11	C or J
{D, S, A, J}	2	D or J

{C, S, A, J}	4	C or J
{H, D, C, S, A}	0	C
{H, D, C, S, J}	7	C or J
{H, D, C, A, J}	9	C or J
{H, D, S, A, J}	0	D or J
{H, C, S, A, J}	2	C or J
{D, C, S, A, J}	4	C or J
{H, D, C, S, A, J}	0	C or J

The algorithm first computes all rows of the table for subsets of size 1, then all rows for subsets of size 2, etc. This is necessary since computing results for a subset of size k requires using results from subsets of size $k - 1$.

Sample Calculations

The following are several calculations that illustrate how each row in the table is computed.

Subset {H} :

number of errors caused by having aspect H in first position = 6
(errors : AH $\succ$ AS, AH $\succ$ JS, JH $\succ$ AS, JH $\succ$ JS, JH $\succ$ AD, JH $\succ$ AC)

store $J(\{H\}) = 6$, with H as optimal last aspect

Subset {H, D} :

cost of having H last : $J(\{D\})$ + newErrors(H after {D}) = 8 + 5 = 13
(new errors : AH $\succ$ AS, AH $\succ$ JS, JH $\succ$ AS, JH $\succ$ JS, JH $\succ$ AC)
cost of having D last : $J(\{H\})$ + newErrors(D after {H}) = 6 + 5 = 11

store $J(\{H, D\}) = 11$, with D as optimal last aspect

Subset {S, A} :

cost of having S last : $J(\{A\})$ + newErrors(S after {A}) = 3 + 0 = 3
cost of having A last : $J(\{S\})$ + newErrors(A after {S}) = 0 + 0 = 0

store $J(\{S, A\}) = 0$, with A as optimal last aspect

Subset {H, D, C} :

cost of having H last : $J(\{D, C\})$ + newErrors(H after {D, C}) = 15 + 4 = 19
cost of having D last : $J(\{H, C\})$ + newErrors(D after {H, C}) = 13 + 4 = 17
cost of having C last : $J(\{H, D\})$ + newErrors(C after {H, D}) = 11 + 4 = 15

store $J(\{H, D, C\}) = 15$, with C as optimal last aspect

Note that we do not compute costs for all 3! = 6 permutations of {H, D,C}. Instead we "try" each aspect in the last position and simply look up the best possible cost for the aspects preceding the last aspect (which was computed during an earlier step).

Subset {H, D, C, S, A, J} :

cost of having H last : $J(\{D, C, S, A, J\})$ + newErrors(H after {D, C, S, A, J}) = 4 + 0 = 4
cost of having D last : $J(\{H, C, S, A, J\})$ + newErrors(D after {H, C, S, A, J}) = 2 + 0 = 2
cost of having C last : $J(\{H, D, S, A, J\})$ + newErrors(C after {H, D, S, A, J}) = 0 + 0 = 0
cost of having S last : $J(\{H, D, C, A, J\})$ + newErrors(S after {H, D, C, A, J}) = 9 + 0 = 9
cost of having A last : $J(\{H, D, C, S, J\})$ + newErrors(A after {H, D, C, S, J}) = 7 + 0 = 7
cost of having J last : $J(\{H, D, C, S, A\})$ + newErrors(J after {H, D, C, S, A}) = 0 + 0 = 0

store $J(\{H, D, C, S, A, J\}) = 0$, with C and J as optimal last aspects

Because $J(\{H, D, C, S, A, J\}) = 0$, it means that there exists an order of aspects that is 100% consistent with the profile preferences

$$AS \succ JS \succ AH \succ AD \succ AC \succ JH \succ HD \succ JC$$

Extracting the Optimal Solutions

To construct the consistent aspect orders, we work backwards starting from the set of all aspects {H, D, C, S, A, J} and seeing which aspects can occur in the last position. For this example, C or J can occur last, i.e., the aspect orders have the following patterns:

$$\{H, D, S, A, J\} \succ C$$
$$\{H, D, C, S, A\} \succ J$$

When aspect C is last, we then consider how to optimally order the remaining aspects that precede C, i.e., {H, D, S, A, J}. Looking up this subset in the table, we find that aspect D or J can occur in the next to last position:

$$\{H, S, A, J\} \succ D \succ C$$
$$\{H, D, S, A\} \succ J \succ C$$

Continuing in this fashion, we can construct all possible consistent aspect orders.

$$S \succ A \succ J \succ H \succ D \succ C$$
$$S \succ A \succ H \succ J \succ D \succ C$$
$$S \succ A \succ H \succ D \succ J \succ C$$
$$S \succ A \succ H \succ D \succ C \succ J$$

Finally, we eliminate redundant aspects. For example, because {A, J} make up a feature, once we know that A is in an order, we do not need J. Similarly, because {S, H, D, C} make up the

feature of "suit," once we know that A, H, and D are in an aspect order, we do not need C. Based on these relationships, we eliminate J and C to get the unique order:

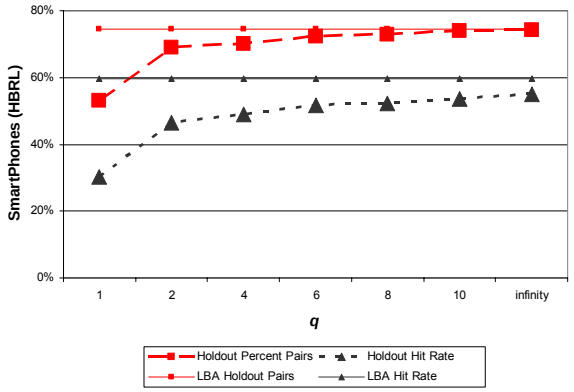
$$S \succ A \succ H \succ D$$

This aspect order reproduces the profile order with zero error.

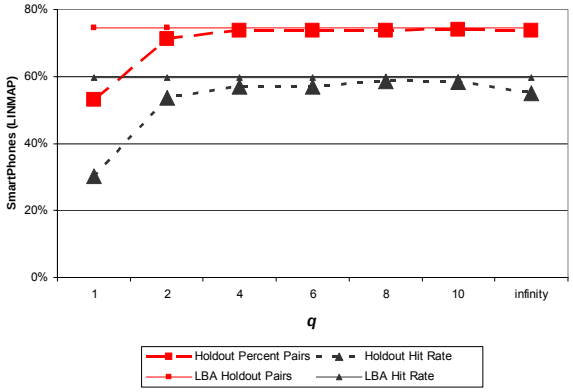
Supplemental Appendix 2: Predictive Ability of *q*-compensatory Benchmarks

In the text we plot holdout predictive ability (percent pairs predicted; hit rate) for HBRL(q) as estimated on the SmartPhone data. That plot is repeated here. We also plot holdout predictive ability for LINMAP(q) and for the Lenk, et. al. (1996) computer data.

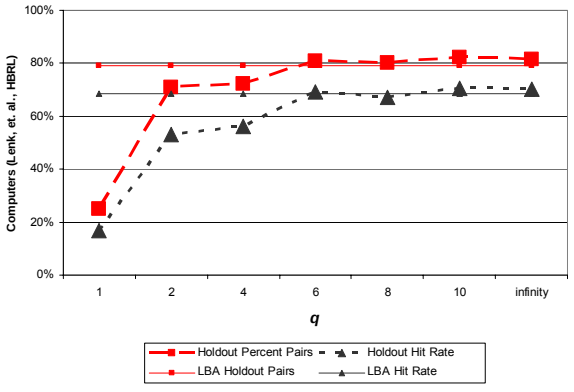
Figure SA2
Holdout Predictive Ability as a Function of *q*



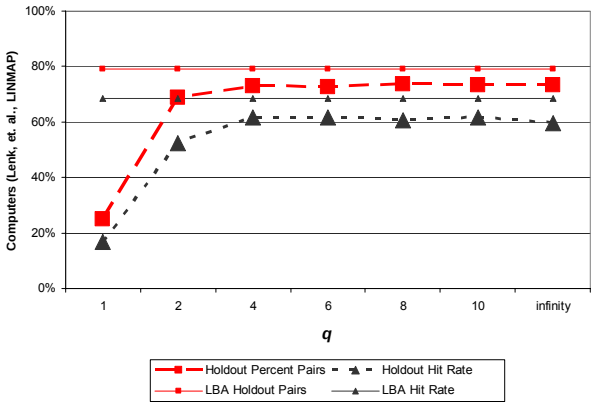
(a) SmartPhones, HBRL



(b) SmartPhones, LINMAP



(c) Computers, HBRL



(d) Computers, LINMAP

Supplemental Appendix 3: Monte Carlo Experiments

For simplicity of exposition, we report Monte Carlo results for rank-order data only. We expect to obtain qualitatively similar results for consider-then-rank synthetic data. For our generating model, we modify a functional form proposed by Einhorn (1970). 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 n^{th} 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.

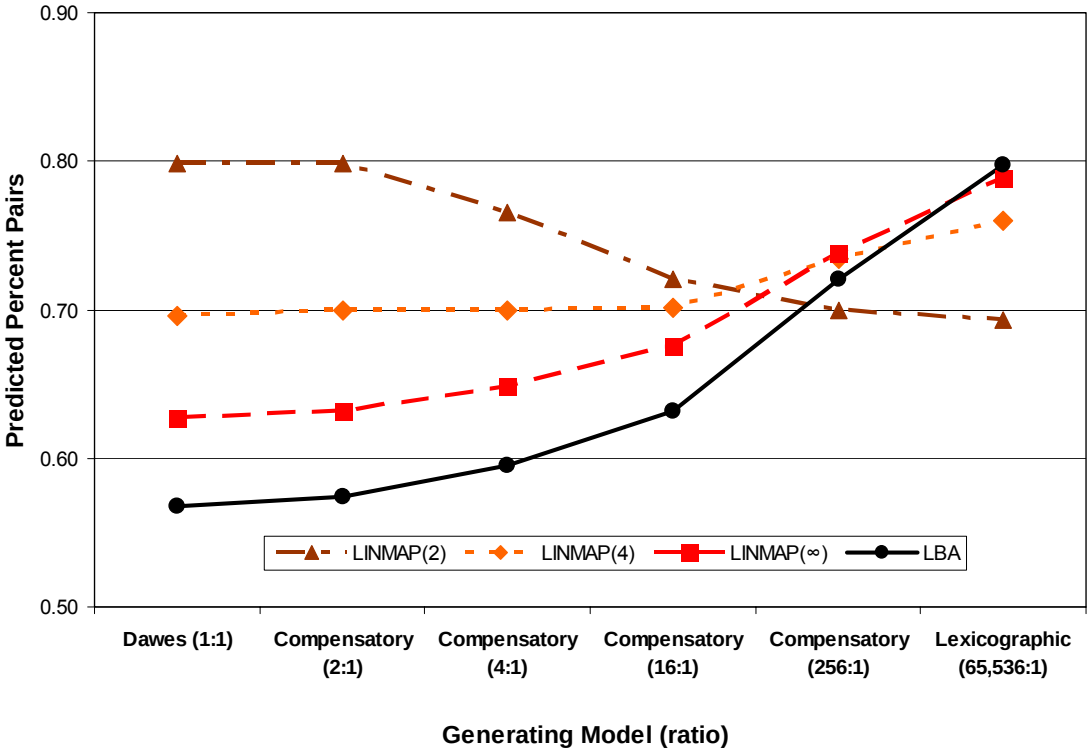
1. For each m, generate $\vec{w}_c$, normalize so w_{nc}'s sum to 1.0. $\vec{w}_c \Rightarrow u_{jc} = \vec{p}'_j \vec{w}_c$.
2. For each c, add error to the true utility: $\tilde{u}_{jc} = u_{jc} + \tilde{\varepsilon}_j$ *where* $\tilde{\varepsilon}_j \sim N(0, e)$, $e = 0.2, 0.4$.
3. Given $\{\tilde{u}_{kc}\}$, generate a rank order of 32 cards for respondent c. Repeat for all m.

For each respondent, we use either the greedoid-based dynamic program to estimate an LBA aspect order or LINMAP(q) to estimate partworths. (As Supplemental Appendix 2 indicates, we obtain very similar predictions with HBRL(q) and LINMAP(q). LINMAP(q) is much more efficient for simulations.) Estimated partworths imply a rank-order of 32 profiles from a $4^3 2^4$ design. The comparison statistic is the percent of ordered pairs of profiles predicted from the estimated model that are consistent with the true model. The results are shown in Figure SA3. 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 ho-

mogeneous respondents, constraints help significantly for low *m*. The unconstrained compensatory model, LINMAP(∞) may over-fit the data for low *m*. We expect this to be mitigated in the SmartPhone and computer empirical studies. Finally, we see that $q = 4$ is a reasonable discriminator vs. LBA because LBA provides superior predictions when respondents are truly lexicographic and $q = 4$ provides superior predictions when respondents are truly compensatory. To obtain even more discriminate ability, we might have chosen $q = 2$ for illustrative purposes. We did not do so because we felt it was too constrained to be realistic. However, should the reader wish to make comparisons to $q = 2$ (or other q's), the data are presented in Figure SA2.

Figure SA3
Results of the Monte Carlo Experiments



As a test of robustness, we also generated data for heterogeneous respondents. Partworths were generated randomly and then raised to a power (m_{hetero}).[1] The curves are qualitatively similar but cross slightly earlier. Predictive ability is reduced slightly for all models. We encourage readers to explore alternative simulations.

[1] m_{hetero} affects the heterogeneous model differently than *m* affects the homogeneous model. For example, for $m_{hetero} = 2$, the geometric mean of the ratio of the maximum partworth to the minimum partworth is the order of 200; for $m_{hetero} = 4$ it is the order of 50,000; and for $m_{hetero} = 8$ it is the order of 1 billion. The arithmetic means are considerably higher. We found the simulations with homogeneous respondents to be easier to interpret.